Introduction to Mobile Application Development on Android

Mobile Computing & Internet of Things
LTAT.06.009

Jakob Mass
jakob.mass [at] ut.ee
What is Android?

• An open-source mobile platform
  • Not just for phones
    • TV, Watch, Smart Home Hub, Car Infotainment, Handheld GPS, ...

• A Linux-based, multiprocess, multithreaded OS
  • Each application is a different user
  • By default, every app runs in its own Linux process.
  • Each process has its own virtual machine, so an app's code runs in isolation from other apps.

• Libraries & Support tools
  • IDE, testing frameworks, emulators

Image: Wikimedia Commons user Wikideas1. The Android robot is reproduced or modified from work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License
History of Android

- Android Inc. was founded in 2003
  - First efforts targeted digital cameras
- Acquired by Google in 2005
- Google, Open Handset Alliance
  - 84 technology companies
  - Commitment to openness, shared vision and concrete plans
- Today it’s the most widely used OS overall (37.9 % share)[1]

Writing Application Code

• You can write native code for Android in Java, Kotlin or C/C++.
  • (Not taking into account 3rd party options)
  • Today, Java is the most used language for Android
  • Kotlin is the promoted language by Google

• Android Projects also involve XML-based UI and component descriptions

• Cross-platform tools such as Flutter
Android Software stack
Linux Kernel

• Foundation of the Android platform
• Handles thread and low-level memory management
• Provides key security features:
  • User-based permissions model
  • Process isolation
  • Application Sandbox
• E.g. file permission management:
  • one user cannot read/modify another user's files
  • Thus, one application cannot see the files created by another application, except when explicitly shared by the developer
• Rooting
  • Only the kernel and a small subset of the core applications run with root permissions
  • It’s possible to grant root access to applications, giving full access to system files, applications and all application data

https://source.android.com/security/overview/kernel-security
Hardware Abstraction Layer (HAL)

- Provides interfaces for accessing device hardware capabilities from the higher-level Java API.
- HAL is divided into various modules/libraries: Camera, Bluetooth, etc., ..

https://source.android.com/devices/architecture/hal
Android Runtime

- Provides an environment to host applications
- Applications run using **ART** (Android Runtime)
  - Since Android v5.0 (API level 21)
  - Before, **Dalvik** was used
- Dalvik Executable (DEX) format
  - bytecode format designed specially for Android
- ART takes care of:
  - executing Dex bytecode specification

Check out “Comparative Analysis of Mobile App Reverse Engineering Methods on Dalvik and ART” by Na et al.
Check out "Comparative Analysis of Mobile App Reverse Engineering Methods on Dalvik and ART" by Na et al.
Why Android/Dalvik Runtime?

• The runtime is optimized specifically for mobile applications

• Runs multiple VMs efficiently
  • Each app runs in its own process and with its own instance of ART

• Minimal memory footprint

• Relies on Linux kernel for threading and low-level memory management
Android Software Stack

- Native C/C++ Libraries

- Native C/C++ Libraries
  - Webkit
  - OpenMAX AL
  - Libc
  - Media Framework
  - OpenGL ES
  - ...

- Android Runtime
  - Android Runtime (ART)
  - Core Libraries

- Hardware Abstraction Layer (HAL)
  - Audio
  - Bluetooth
  - Camera
  - Sensors
  - ...

- Linux Kernel
Java API & System Apps

https://developer.android.com/guide/platform
Different API versions..

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.3 - 2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>0.3%</td>
</tr>
<tr>
<td>4.0.3 - 4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>0.3%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>1.2%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>1.5%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>0.5%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>6.9%</td>
</tr>
<tr>
<td>5.0</td>
<td>Lollipop</td>
<td>21</td>
<td>3.0%</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>22</td>
<td>11.5%</td>
</tr>
<tr>
<td>6.0</td>
<td>Marshmallow</td>
<td>23</td>
<td>16.9%</td>
</tr>
<tr>
<td>7.0</td>
<td>Nougat</td>
<td>24</td>
<td>11.4%</td>
</tr>
<tr>
<td>7.1</td>
<td></td>
<td>25</td>
<td>7.8%</td>
</tr>
<tr>
<td>8.0</td>
<td>Oreo</td>
<td>26</td>
<td>12.9%</td>
</tr>
<tr>
<td>8.1</td>
<td></td>
<td>27</td>
<td>15.4%</td>
</tr>
<tr>
<td>9</td>
<td>Pie</td>
<td>28</td>
<td>10.4%</td>
</tr>
</tbody>
</table>

https://developer.android.com/about/dashboards/ (September 2019)
<table>
<thead>
<tr>
<th>Android Platform Version</th>
<th>API Level</th>
<th>Cumulative Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0 Ice Cream Sandwich</td>
<td>15</td>
<td>99.6%</td>
</tr>
<tr>
<td>4.1 Jelly Bean</td>
<td>16</td>
<td>98.1%</td>
</tr>
<tr>
<td>4.2 Jelly Bean</td>
<td>17</td>
<td>95.9%</td>
</tr>
<tr>
<td>4.3 Jelly Bean</td>
<td>18</td>
<td>95.3%</td>
</tr>
<tr>
<td>4.4 KitKat</td>
<td>19</td>
<td>85.0%</td>
</tr>
<tr>
<td>5.0 Lollipop</td>
<td>21</td>
<td>80.2%</td>
</tr>
<tr>
<td>5.1 Lollipop</td>
<td>22</td>
<td>62.6%</td>
</tr>
<tr>
<td>6.0 Marshmallow</td>
<td>23</td>
<td>37.1%</td>
</tr>
<tr>
<td>7.0 Nougat</td>
<td>24</td>
<td>14.2%</td>
</tr>
<tr>
<td>7.1 Nougat</td>
<td>25</td>
<td>6.0%</td>
</tr>
<tr>
<td>8.0 Oreo</td>
<td>26</td>
<td>1.1%</td>
</tr>
<tr>
<td>8.1 Oreo</td>
<td>27</td>
<td></td>
</tr>
</tbody>
</table>

Lollipop

**Wireless & Connectivity**
- Multiple SIM Card Support
- Carrier Provisioning

**API Change**
- Deprecated HTTP classes

https://developer.android.com/aboutversions/android-5.1.html
Development Environment & Tools

• Android Studio, IntelliJ IDEA-based set of software tools
• Single, unified environment for developing Android
  • Download manager for SDK versions
  • Emulator
  • Performance profiling
  • Visual layout editing
  • Testing tools
  • Code templates
  • …

(Formerly, Eclipse was used)

https://developer.android.com/studio/intro/
Development Environment & Tools

https://developer.android.com/studio/intro/
Generally, we work with the build.gradle files only, especially the module-level build.gradle file!

Projects in Android Studio

• Applications are created as **Gradle**-based projects

• Gradle is a build automation-tool, it takes care of software development routines:
  • compiling the source code
  • executing tests
  • download and configuration of dependencies or other libraries
  • packaging the application and additional files
  • Installing the application to a physical/virtual device and running it there
  • ..

https://gradle.org/
https://developer.android.com/studio/build
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.example.myapplication"
        minSdkVersion 21
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
}
Application Manifest

• Each project must declare an `AndroidManifest.xml` file
• Describes essential information about the app to the build tools, the Android OS, and Google Play
• It declares for instance:
  • The permissions that the app wishes to use
  • Capabilities such as supported device configurations
  • Metadata, such as the application package name
  • The components of the app (Activities, Services, BroadcastReceiver, ..), and the corresponding Java (Kotlin) class names

https://developer.android.com/guide/topics/manifest/manifest-intro
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">
    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.SEND_SMS"/>
</manifest>
Security in Android

- As mentioned earlier, follows standard Linux guidelines
- Each application runs in its own process
- Process permissions are enforced at user level and group IDs are assigned to processes
- Finer grained permissions are then granted (revoked) per operations
- Apps declare permissions in the manifest

https://developer.android.com/guide/topics/permissions/overview
Application Design Philosophy

• Applications should be
  • Lightweight
    • Resources are constrained: if your app consumes too much RAM while in the background, it will get killed by the OS quickly!
  • Responsive
    • The framework strongly discourages freeze-ups
  • Secure, respectful of privacy
    • Declare and ask for only the permissions you need, when you need them!
  • Seamless
    • Usability is key, the framework gives you lots of tools to support accessibility & localization options and storing, persisting data
Application priorities, process states

• Applications have limited control over their own lifecycles

• Recall that applications run in separate processes, each one in a separate instance of the ART virtual machine

• Memory and process management is handled by the runtime and kernel
  • Runtime may kill some services in the background
  • Being aware of application states & priorities is critical
Process lifecycle, state hierarchy

1. Foreground process (active process)
   • Few simultaneously

2. Visible process
   • Activity that is visible to the user on-screen but not in the foreground. E.g., if the foreground Activity is displayed as a dialog that allows the previous Activity to be seen behind it.

3. Service process
   • such as background network data upload or download

4. Cached process

Recap

- Android framework is based on Linux Kernel
- Native Android apps are built
  - Kotlin / Java
  - Gradle
- Backwards compatibility
- Process lifecycle and application states
  - Important to understand, we will see more in next lecture
This week in lab session

• Setting up development tools & Hello World
• Introduction to UI creation
• Resource management

• Please download ( & install ) Android Studio & Emulator before the session!
Next lecture

• Kotlin 101
• Android activity lifecycle
• Managing UI
  • Overview of UI components
  • Resource files