Introduction to Android & Android Studio

Mobile Application Development
LTAT.06.021

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 - a different user
  • By default, every app runs in its own 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
  • Lead by Andy Rubin
  • 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
• Most widely used OS overall (44.6 %) [1]
• 3bn monthly active devices in 2021 [2]

Writing Application Code

• You can write native code for Android in Java, Kotlin or C/C++.
  • (Not taking into account 3rd party options, e.g. Xamarin)
  • Java used to be the main language
    • until 2019
  • Today, Kotlin is the language promoted by Google
• Android Projects also involve XML-based UI and component descriptions
Android Software stack

Image: https://developer.android.com/guide/platform/
https://source.android.com/devices/architecture
Foundation - Linux Kernel

- Handles:
  - Threading and low-level memory management
  - Process management
  - Filesystem management

- Device manufacturers can develop drivers for well-known kernel

- Provides key security features:
  - User-based permissions model & resource isolation
  - Process isolation, application sandboxing
  - Secure interprocess communication

- E.g. file permission management:
  - one user cannot read/modify another user's files

https://source.android.com/security/overview/kernel-security
Application isolation

• Each application runs in its own sandbox
  • it’s own user, in its own process
  • This is the basis of the Android security model
  • Applies to all software above the kernel

• Thus, one app cannot see the files created by another app
  • 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
    • For development / debugging purposes only

https://source.android.com/security/app-sandbox
https://source.android.com/security/overview/kernel-security.html#rooting-devices
Hardware Abstraction Layer

- Provides consistent interfaces for accessing device hardware capabilities from the higher-level Java API.
- Hardware vendors implement HAL modules which use the Kernel-level drivers.
- HAL is divided into various modules/libraries: Camera, Bluetooth, Biometrics etc.,..

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

• Applications run using **ART** (Android Runtime)
  • Provides an environment to host applications (similar to JVM)
• ART executes Dalvik Executable (DEX) format
  • bytecode format designed specially for Android
  • ART replaced the *Dalvik* runtime from Android v5.0 (API level 21)
• Handles Garbage Collection, AOT Compilation, etc

https://developer.android.com/guide/platform/#art
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, optimized GC

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

• Native C/C++ Libraries

- Webkit
- OpenMAX AL
- Libc
- Media Framework
- OpenGL ES
- ...

• Core libraries such as SSL, libc
• Media & Graphics libraries, database
• Android provides Java APIs to access these
Java API & System Apps

- The full Android feature set is exposed through Java APIs.
  - Same APIs are used by system apps

https://developer.android.com/guide/platform
API Versions
<table>
<thead>
<tr>
<th>ANDROID PLATFORM VERSION</th>
<th>API LEVEL</th>
<th>CUMULATIVE DISTRIBUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Jelly Bean</td>
<td>16</td>
<td>99.9%</td>
</tr>
<tr>
<td>4.2 Jelly Bean</td>
<td>17</td>
<td>99.7%</td>
</tr>
<tr>
<td>4.3 Jelly Bean</td>
<td>18</td>
<td>99.7%</td>
</tr>
<tr>
<td>4.4 KitKat</td>
<td>19</td>
<td>99.7%</td>
</tr>
<tr>
<td>5.0 Lollipop</td>
<td>21</td>
<td>98.8%</td>
</tr>
<tr>
<td>5.1 Lollipop</td>
<td>22</td>
<td>98.4%</td>
</tr>
<tr>
<td>6.0 Marshmallow</td>
<td>23</td>
<td>96.2%</td>
</tr>
<tr>
<td>7.0 Nougat</td>
<td>24</td>
<td>92.7%</td>
</tr>
<tr>
<td>7.1 Nougat</td>
<td>25</td>
<td>90.4%</td>
</tr>
<tr>
<td>8.0 Oreo</td>
<td>26</td>
<td>88.2%</td>
</tr>
<tr>
<td>8.1 Oreo</td>
<td>27</td>
<td>85.2%</td>
</tr>
<tr>
<td>9.0 Pie</td>
<td>28</td>
<td>77.3%</td>
</tr>
<tr>
<td>10. Q</td>
<td>29</td>
<td>62.8%</td>
</tr>
<tr>
<td>11. R</td>
<td>30</td>
<td>40.5%</td>
</tr>
<tr>
<td>12. S</td>
<td>31</td>
<td>13.5%</td>
</tr>
</tbody>
</table>

https://apilevels.com/
Backwards Compatibility

• Currently active versions in the market fairly fragmented
• Can’t just go for latest version unless tiny target audience is OK for us
• Android platform can provide some backwards-compatibility
  • E.g. running on old device, but targeting new API levels, modern themes are applied automatically
  • Android Support Libraries
• But New APIs may not be available on old devices
  • Recommended to check device version, provides basic/core functionality without relying on newest APIs
Development tools
Android Studio

• 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 - Emulator

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

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

• E.g. to build your project from the command-line, you could use:
  $ ./gradlew build

[Links]
https://gradle.org/
https://developer.android.com/studio/build
Build Process

- Resources, code, AndroidManifest.xml are packaged into an **.apk** file
  - (apk is signed with debug/ release key)
- .apk is deployed to device and installed
- .apk file is basically an archive

https://developer.android.com/studio/build
Recap

• Android framework is based on Linux Kernel
• SDK Versions Management & Backwards compatibility
• Native Android apps are built using:
  • Kotlin / Java
  • Gradle
This week

• Labs start on Wednesday!
• Bring your laptop!
• Download & Install Android Studio **before** the lab!
  • Follow the guide here: https://courses.cs.ut.ee/2022/MAD/fall/Main/SettingUpAndroidStudio