Introduction to Android & Android Studio

Mobile Computing & Internet of Things
LTAT.06.009

Jakob Mass
jakob.mass at ut.ee
Welcome to the course

• https://cs.ut.ee/reg

• https://courses.cs.ut.ee/2020/MCIoT/fall

• Android Studio installed already?
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
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, e.g. Xamarin)  
  • 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
Android Software stack

Image: https://developer.android.com/guide/platform/
https://source.android.com/devices/architecture
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 sandboxing
• 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

[Link to Developer's Guide](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
• 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
# Java API & System Apps

## System Apps
- Dialer
- Email
- Calendar
- Camera
- ...

## Java API Framework
- Content Providers
- View System
- Managers
  - Activity
  - Location
  - Package
  - Notification
  - Resource
  - Telephony
  - Window

API Versions
Pie chart: [https://developer.android.com/about/dashboards/](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>99.8%</td>
</tr>
<tr>
<td>4.1 Jelly Bean</td>
<td>16</td>
<td>99.2%</td>
</tr>
<tr>
<td>4.2 Jelly Bean</td>
<td>17</td>
<td>98.1%</td>
</tr>
<tr>
<td>4.3 Jelly Bean</td>
<td>18</td>
<td>98.4%</td>
</tr>
<tr>
<td>4.4 KitKat</td>
<td>19</td>
<td>94.1%</td>
</tr>
<tr>
<td>5.0 Lollipop</td>
<td>21</td>
<td>92.3%</td>
</tr>
<tr>
<td>5.1 Lollipop</td>
<td>22</td>
<td>84.9%</td>
</tr>
<tr>
<td>6.0 Marshmallow</td>
<td>23</td>
<td>73.7%</td>
</tr>
<tr>
<td>7.0 Nougat</td>
<td>24</td>
<td>66.2%</td>
</tr>
<tr>
<td>7.1 Nougat</td>
<td>25</td>
<td>60.8%</td>
</tr>
<tr>
<td>8.0 Oreo</td>
<td>26</td>
<td>53.5%</td>
</tr>
<tr>
<td>8.1 Oreo</td>
<td>27</td>
<td>39.5%</td>
</tr>
<tr>
<td>9.0 Pie</td>
<td>28</td>
<td>8.2%</td>
</tr>
<tr>
<td>10. Android 10</td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>
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 - Emulator

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

[https://gradle.org/](https://gradle.org/)
Recap

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