RE Android

Základy reverzného inžinierstva

Boot proces

1. Power On / Boot ROM

CPU začne spracovávať prvé inštrukcie z hardvérom preddefinovaného miesta.

2. Bootloader

(normal, recovery, download mode) (fastboot oem unlock)

Hardvérová identifikácia, načítanie jadra OS do pamäte, ...

3. Jadro

Inicializácia systémových rutín, správa pamäte, súborového systému, prerušenia, atď. Alokovanie **init.rc** skriptu.

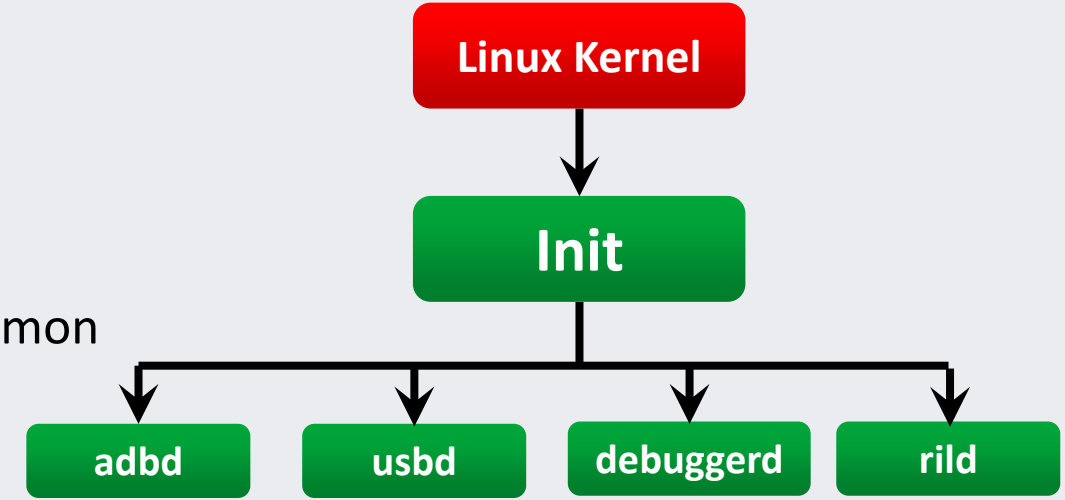
Init

Android Init Language

(AOSP/system/core/init/...)

Init spúšťa Linux deamons:

- Android Debug Bridge Deamon
- USB Deamon
- Debugger Deamon
- Radio Interface Layer Deamon
- ...

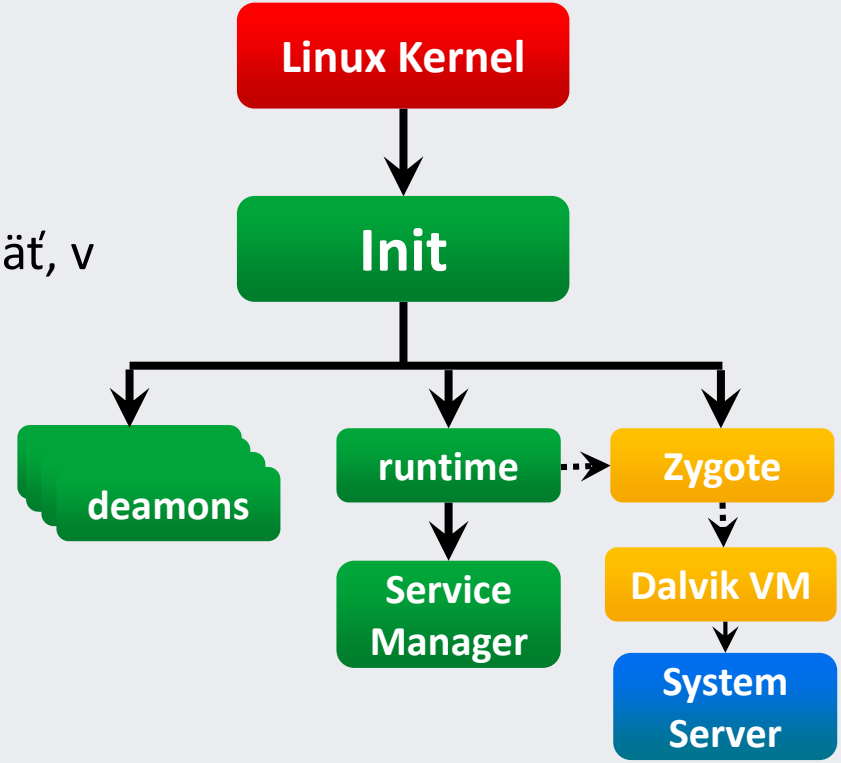


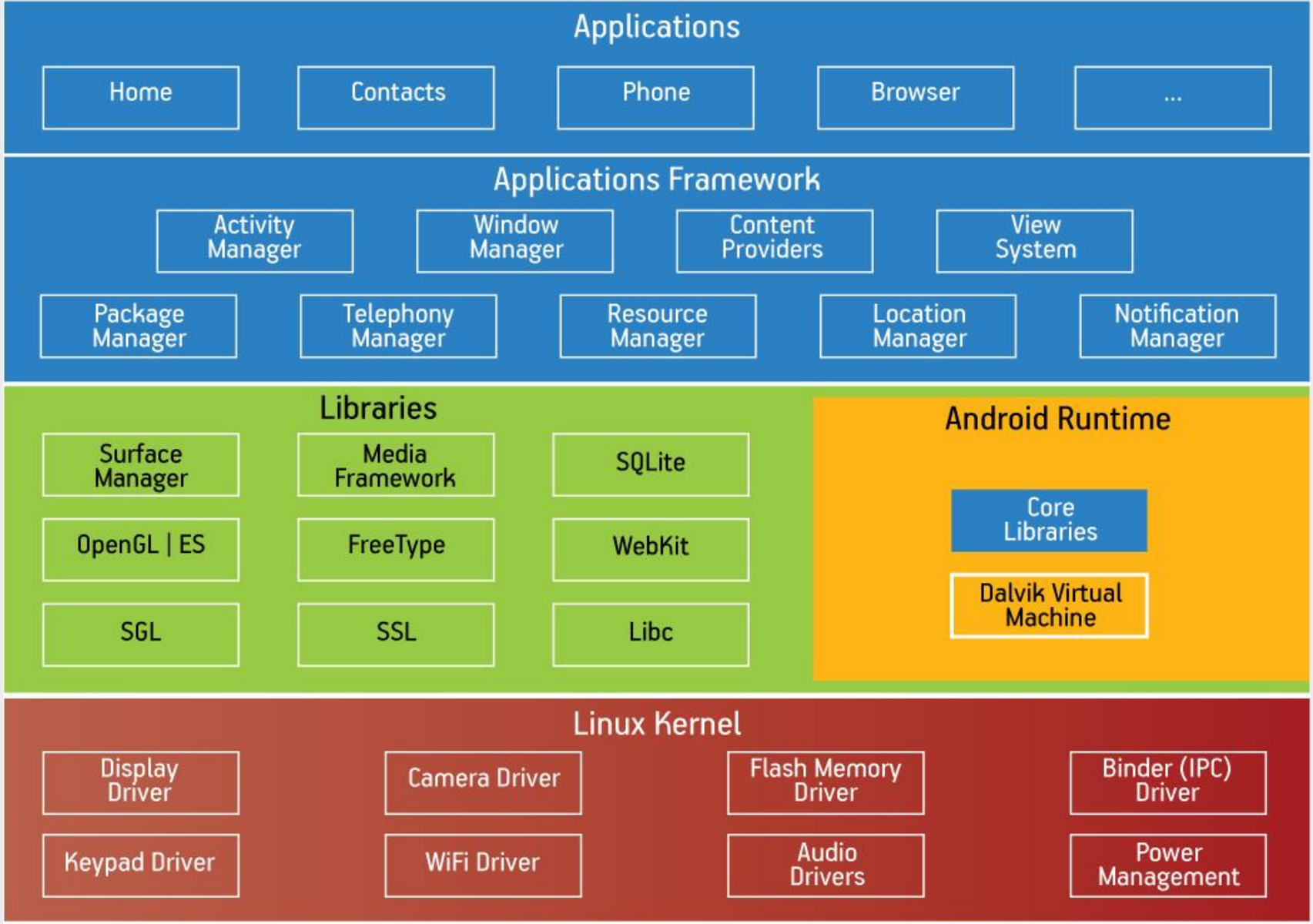
Skript sa nachádza v root adresáry „/init.rc“

Zygote

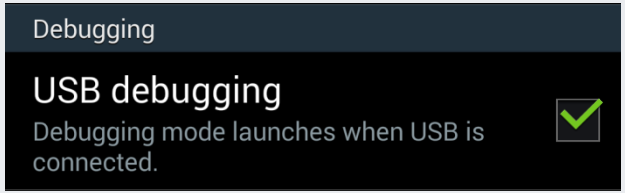
Spustenie prvého procesu **Zygote.**

- Z neho budú tvorené (fork) všetky ostatné procesy a zdielať jeho pamäť, v ktorej sa nachádza:
  - Inštancia Dalvik VM
  - Pred nahraté knižnice (AOSP/framework/base/preloaded-classes)




Applications
Home
Contacts
Phone
Browser
...
Applications Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
Libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

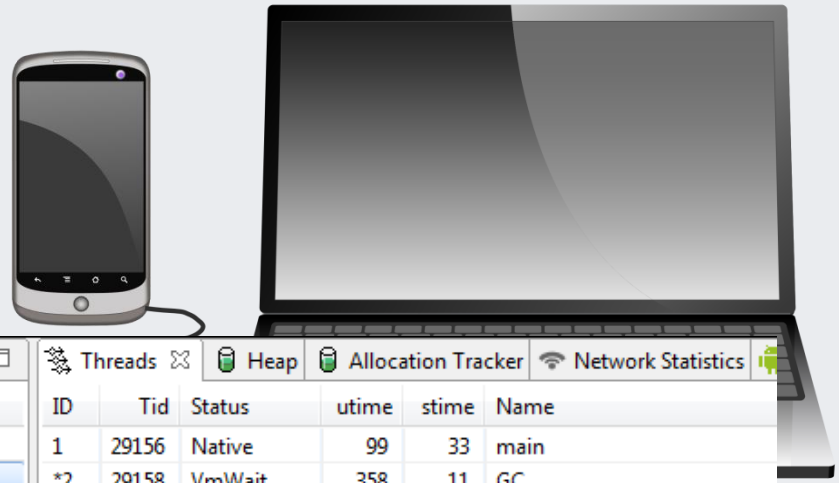
Android Debug Bridge



- AndroidSDK/platform-tools/adb devices
- adb shell
  - ls –la
  - cat init.rc
  - ps
  - cat /proc/sys/kernel/ostype {osrelease, version}
  - dumpsys package “package”
  - dumpsys activity services | grep “package”
  - cd /system/app {/data/app; /data/data; /sdcard}

Android Device Monitor

- AndroidSDK/tools/monitor



Devices

Name			
samsung-gt_i9100-0009f52e56ca8f	Online		4.2.2, debug
system_process	29156		8600 / 8700
com.android.systemui	30937		8601
android.process.media	31030		8602
com.android.defcontainer	31087		8614
com.android.gallery3d	31115		8607
com.tmobile.thememanager	31145		8618
com.android.voicedialer	31159		8620
com.google.android.googlequicksearchbox	31174		8621
com.android.inputmethod.latin	31331		8625
com.android.vending	31393		8603
com.android.smspush	31452		8608
com.google.android.gms	31471		8606
com.cyanogenmod.trebuchet	31493		8605
com.google.process.gapps	31487		8609
com.android.phone	31548		8604
com.google.process.location	31579		8611

Threads | Heap | Allocation Tracker | Network Statistics

ID	Tid	Status	utime	stime	Name
1	29156	Native	99	33	main
*2	29158	VmWait	358	11	GC
*3	29159	VmWait	0	0	Signal Catcher
*4	29160	Runnable	168	339	JDWP
*5	29161	VmWait	177	98	Compiler
*6	29164	Wait	2	0	ReferenceQueueDaemon
*7	29165	Wait	40	10	FinalizerDaemon
*8	29166	Wait	0	0	FinalizerWatchdogDaemon
9	29167	Native	82	29	Binder_1
10	29168	Native	96	25	Binder_2
11	29174	Native	1838	922	android.server.ServerThread
12	29173	Native	446	1232	SensorService
13	29175	Native	146	23	UI
14	29176	Native	53	13	WindowManager
15	29177	Native	104	138	ActivityManager
16	29179	TimedWait	5	4	ProcessStats

Základné operácie

Inštalácia

- adb install HelloWorld.apk

Spustenie

- adb shell am start com.test.helloworld/.HelloWorldActivity

Umiestnenie v systéme

- adb shell dumpsys package com.test.helloworld

Stiahnutie do pc

- adb pull /data/app/com.test.helloworld-1.apk ./downloaded.apk

Odinštalácia

- adb uninstall com.test.helloworld

Aplikácie – vývojové prostredie

- **Android SDK**

  http://developer.android.com/sdk/installing/index.html?pkg=tools
- **Java JDK**

  http://www.oracle.com/technetwork/java/javase/downloads/index.html
- **Apache Ant™**

  http://ant.apache.org/bindownload.cgi
- ***Android NDK***

  https://developer.android.com/tools/sdk/ndk/index.html

IDE

- Textový editor + command line
- Android Studio
- Eclipse

Aplikácie – projekt

- android create project --target 1 --name HelloWorld --path .\HelloWorld --activity HelloWorldActivity --package com.test.helloworld
- ant debug

HelloWorldActivity.java

```
package com.hapo.helloworld;

import android.app.Activity;
import android.os.Bundle;

import android.util.Log;

public class HelloWorldActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Log.i("info", "Hello ZRI!");
    }
}
```

Aplikácie

Android aplikácie sú distribuované formou jedného súboru s koncovkou **.apk**.

Súbor .apk je komprimovaný súborovým formátom zip.

Obsahuje:

- Classes.dex
- AndroidManifest.xml
- Resources
- META-INF

- Natívne knižnice
- Assets (prídavne súbory)

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.helloworld"
    android:versionCode="1"
    android:versionName="1.0">

  <application android:label="@string/app_name">
    <activity android:name=".HelloWorldActivity"
          android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

AndroidManifest.xml – Internet permission

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.helloworld"
    android:versionCode="1"
    android:versionName="1.0">
<uses-permission android:name="android.permission.INTERNET">
  <application android:label="@string/app_name">
    <activity android:name=".HelloWorldActivity"
          android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Permissions

Prídavné oprávnenia ktoré aplikácia vyžaduje od používateľa v čase inštalácie.

1. android.permission.SEND_SMS (RECEIVE_SMS)
2. android.permission.SYSTEM_ALERT_WINDOW
3. com.android.browser.permission.READ_HISTORY_BOOKMARKS (WRITE)
4. android.permission.READ_CONTACTS (WRITE)
5. android.permission.READ_CALENDAR (WRITE)
6. android.permission.CALL_PHONE
7. android.permission.READ_LOGS
8. android.permission.ACCESS_FINE_LOCATION
9. android.permission.GET_TASKS
10. android.permission.RECEIVE_BOOT_COMPLETE
11. android.permission.CHANGE_WIFI_STATE

Permissions

Viažu sa na ne konkrétne API

android.permission.SEND_SMS

sendMultipartTextMessage(..)

sendTextMessage(..)

android.permission.READ_PHONE_STATE

getDeviceId()

getLine1Number()

getSimSerialNumber()

getVoiceMailNumber()

Permissions - INTERNET

AndroidManifest.xml

```
<user-permission android:name=”android.permission.INTERNET” />
```

AOSP/frameworks/base/data/etc/platform.xml

```
<permission name="android.permission.INTERNET" >
<group gid="inet" /> </permission>
```

AOSP/system/core/include/private/android_filesystem_config.h

```
#define AID_INET                3003        /* can create AF_INET and AF_INET6 sockets */
static const struct android_id_info android_ids[] = {
        {       “inet”,         AID_INET, },
```

ALK/net/ipv4/af_inet.c (Android Linux Kernel)

```
static inline int current_has_network(void){
        return in_egroup_p(AID_INET) || capable(CAP_NET_RAW);}
```

Analýza na úrovni permissions

Zlý spôsob

Priamy prístup k službe

Dobrý spôsob

Využitie existujúcich, nato zhotovených, aplikácií

Zlý spôsob

android.permission.SEND_SMS (silently send sms)

Dobrý spôsob

```
Uri smsNumber = Uri.parse(“sms:111233344456”);
Intent intent = net Intent(Intent.ACTION_VIEW);
intent.setData(smsNumber);
intent.putExtra(Intent.EXTRA_TEXT, “Hello”);
super.startActivity(intent);                    //(just open default sms app)
```

classes.dex

Súbor classes.dex obsahuje program aplikácie. Napísaný je v jazyku smali pomocou pseudo-inetrukcií, ktoré Dalvik VM vykonáva.

Formát súboru pozostáva:

Hlavička

Sekcie:

- Strings
- Types
- Prototypes
- Fields
- Methods
- Classes
- Data

SMALI jazyk

Obsahuje pseudo-inštrukcie pre virtuálny stroj Dalvik (**Register-Based VM**).

java source code ----------> .smali <----------> .dex

Príklad: debug výpis

const-string v0, "info"

const-string v1, "Hello World!"

invoke-static {v0, v1}, Landroid/util/Log;->i(Ljava/lang/String;Ljava/lang/String;)I

Java bytecode vs. Dalvik bytecode

```
public class Demo {
    private static final char[] DATA = {
        'A','m','b','e','r',
        ' ','u','s','e','s',' ',
        'A','n','d','r','o','i','d'
    };
}
```

Java

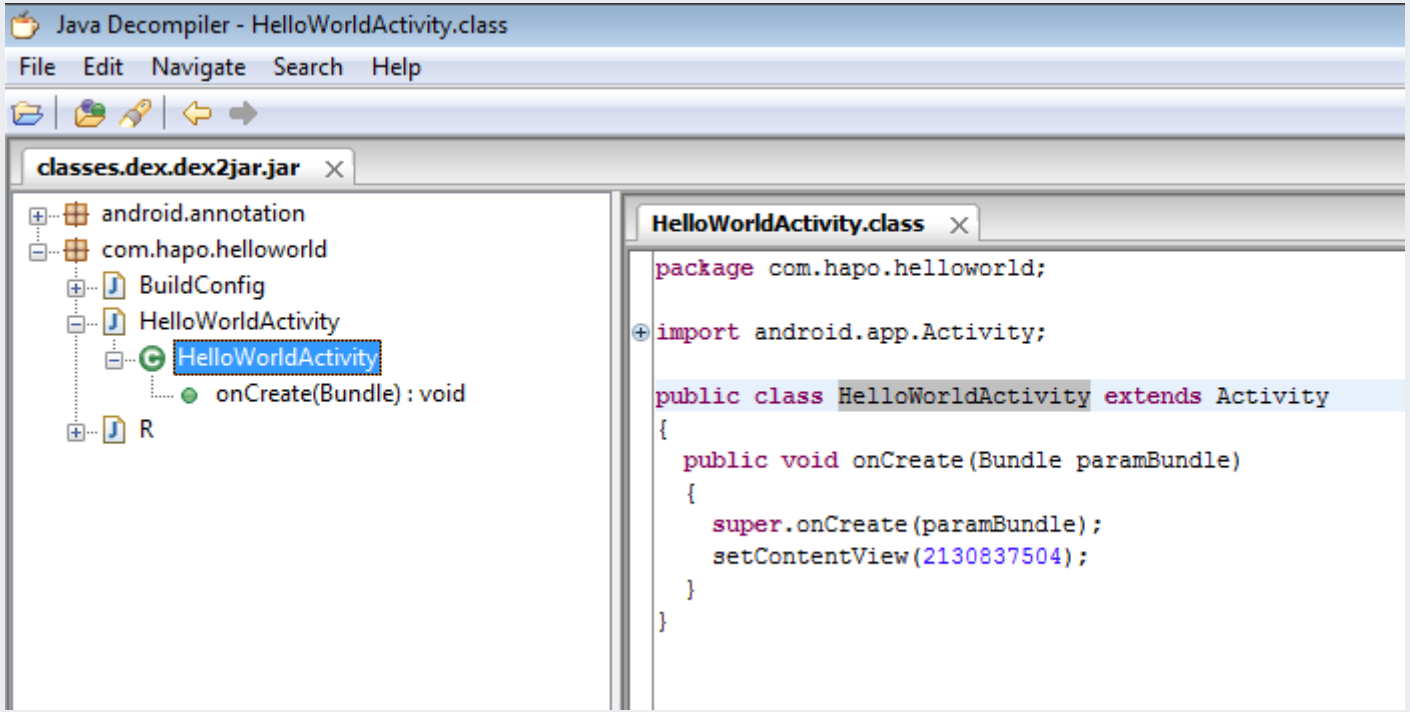
```
  0: bipush  18
  2: newarray char
  4: dup
  5: iconst_0
  6: bipush  65
  8: castore
    ...
101: bipush  17
103: bipush  100
105: castore
106: putstatic #2; // DATA
109: return
```

Dalvik

```
|0000: const/16 v0, #int 18
|0002: new-array v0, v0, [C
|0004: fill-array-data v0,
          0000000a
|0007: sput-object v0,
          LDemo;.DATA:[C
|0009: return-void
|000a: array-data (22 units)
```

dex -> jar

Dekompilácia do čitateľnej formy – Java syntax



Aplikácie – interná reprezentácia

Android aplikácia beží v bezpečnom sandbox prostredí

Android aplikácia je Linux proces

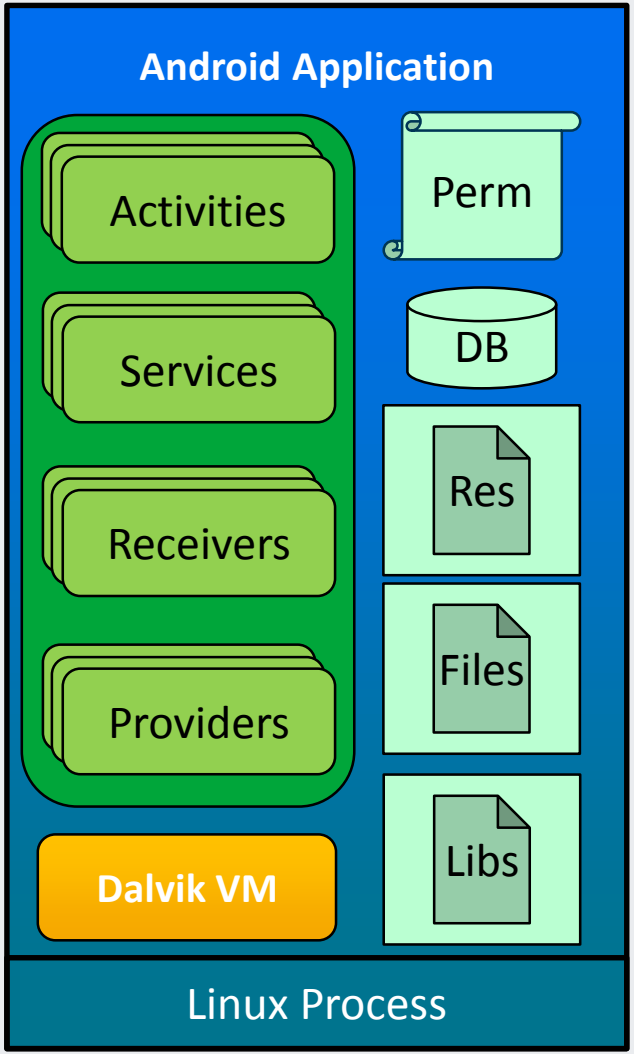
Nesie v sebe inštanciu Dalvik VM

Disponuje so spustiteľnými / funkčnými triedami

- **Activities** – grafické používateľské rozhranie
- **Services** – kód bežiaci na pozadí
- **Receivers** – odchytávanie udalostí
- **Providers –** správa dát

Prídavné oprávnenia

Databáza, resources, súbory, knižnice



Sandboxing

Aplikácie – interná reprezentácia

Každá aplikácia:

- zodpovedá jedinečnému používateľovi na najnižšej linux úrovni. Obdrží svoje UID a GIU.
- Zároveň je samostatným procesom, ktorý dedí po Zygote.
- Prevažná časť jej kódu je interpretovaná virtuálnym strojom Dalvik.
- Zvyšná časť je písaná v C/C++ a kompilovaná do ARM assembleru
- Dalvik VM spracováva programy napísané v jazyku **smali**.

Android NDK – Native Development Kit

Nástroje: Android NDK + Cygwin

```
jclass Java_smsManager = env->FindClass("android/telephony/SmsManager");
if(Java_smsManager ==NULL)
        LOGI("Java_smsManager = NULL!!!");
else
        LOGI("Java_smsManager OK");


jmethodID Java_smsManager_sendTextMessage = env->GetMethodID(Java_smsManager,
     "sendTextMessage",
     "(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Landroid/app/PendingIntent;Landroid/app/PendingIntent;)V");
if(Java_smsManager_sendTextMessage ==NULL)
        LOGI("Java_smsManager_sendTextMessage = NULL!!!");
else
        LOGI("Java_smsManager_sendTextMessage OK");

jmethodID Java_smsManager_Constructor = env->GetMethodID(Java_smsManager, "<init>", "()V");
if(Java_smsManager_Constructor==NULL)
        LOGI("Java_smsManager_Constructor = NULL!!!");
else
        LOGI("Java_smsManager_Constructor OK");

jobject Java_smsManager_Object = env->NewGlobalRef(env->NewObject(Java_smsManager, Java_smsManager_Constructor));
if(Java_smsManager_Object==NULL)
        LOGI("Java_smsManager_Object = NULL!!!");
else
        LOGI("Java_smsManager_Object OK");


jstring phone = env->NewStringUTF("004219xxcccvvv");
jstring text  = env->NewStringUTF("HI FROM NATIVECODE!");

env->CallVoidMethod(Java_smsManager_Object, Java_smsManager_sendTextMessage, phone, 0, text, 0, 0);
```

EOF