

Android

Explicación de práctica 6

Sistemas Operativos

Facultad de Informática
Universidad Nacional de La Plata

2020



1 Overview

- Principales características
- Etimología e Historia
- Versionado
- Stack
- Procesos e Hilos
- DVM/ART vs. JVM
- Aplicaciones
- Seguridad
- Almacenamiento
- File System
- Licencia

2 Rooting

- Definición
- Bootloader
- Boot.img & initramfs
- Booteo/flasheo y testeo



1 Overview

- Principales características
- Etimología e Historia
- Versionado
- Stack
- Procesos e Hilos
- DVM/ART vs. JVM
- Aplicaciones
- Seguridad
- Almacenamiento
- File System
- Licencia

2 Rooting

- Definición
- Bootloader
- Boot.img & initramfs
- Booteo/flasheo y testeo



- Android corre sobre Linux (v4.4.1).
- Android se aprovecha de Linux para:
 - Abstracción de hardware.
 - Administración de memoria.
 - Administración de CPU.
 - Networking.
 - Seguridad.
- El usuario no nota el Linux subyacente.



- ¿Sueñan los androides con ovejas eléctricas? - Philip K. Dick



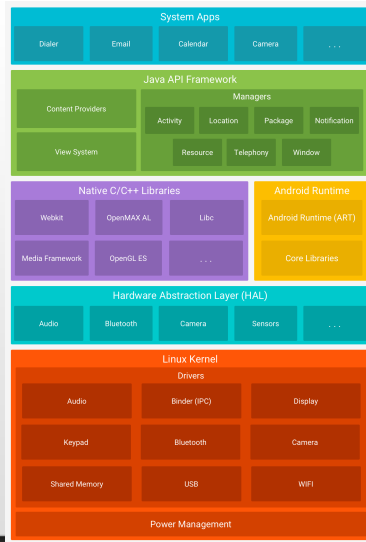
Andy

- Android Inc. → 2003.
- Google → 2005.
- OHA (Open Handset Alliance) → 2007.
- Versión 1.0 (*Apple Pie*) estable → 2008.



- Diversas versiones simbólicas y numéricas.
- Nombre de dulces.
- v4.4 (KitKat), v5.1 (Lollipop), v6.0 (Marshmallow), v7.0 (Nougat), v8.0 (Oreo), v9.0 (P próximamente).





- Componentes: *activity*, *service*, *receiver*, *provider*.
- Cuando se invoca el primer componente de una aplicación se crea un proceso Linux con un único thread (*main/UI thread*).
- Por defecto todos los componentes de una aplicación se ejecutan sobre ese mismo thread del proceso que represente a la aplicación.
- Los componentes pueden ser configurados para que ejecuten sobre diferentes procesos (*android:process*).
- Es posible crear threads dentro de cada proceso → importante para el acceso a la *UI* (*UI toolkit thread unsafe*).

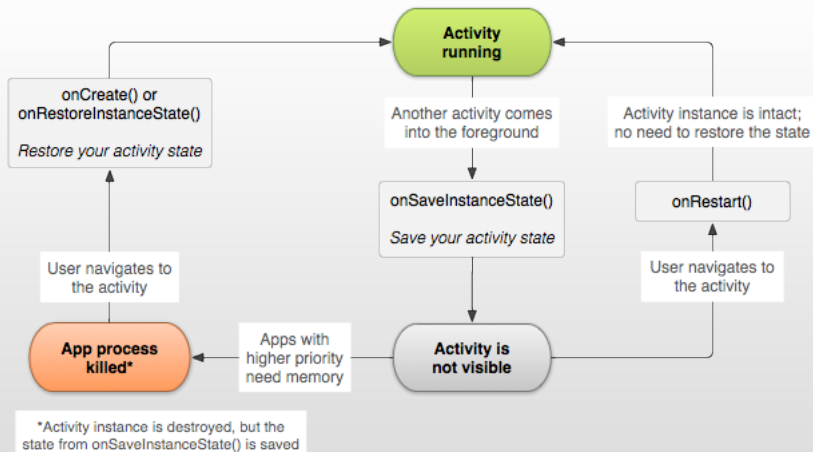


- Android elimina procesos bajo demanda ante la ausencia de memoria → no cuenta con área de intercambio ¹.
- Decisión en base a jerarquía de procesos clasificados en:
 - *Foreground process*
 - *Visible process* → *onPause()*
 - *Service process* → *startService()*
 - *Background process (LRU)* → *onStop()*
 - *Empty process*

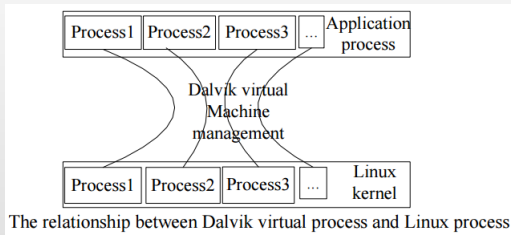
¹<https://zerocredibility.wordpress.com/2009/08/24/why-android-swap-doesnt-make-sense/>



Procesos e Hilos - Seudo swapping



- Individualizada por cada aplicación.
- Cada proceso es un proceso Linux.
- Cada thread es un thread Linux.



DVM/ART	JVM
Máquina basada en registros	Máquina basada en stack
Diseñada para ejecutar sobre poca memoria	Consume más memoria
Ejecuta sus propios bytes codes (<i>.dex</i>) ²	Ejecuta bytes codes Java (<i>.class</i>)
Uni-plataforma → Android	Multi-plataforma
Ejecutable → <i>.apk</i>	Ejecutable → <i>.jar</i>

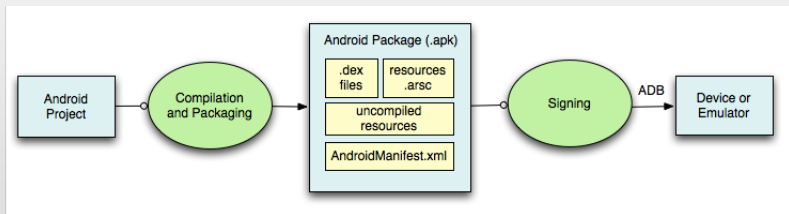
Mejoras de *ART* [ref]:

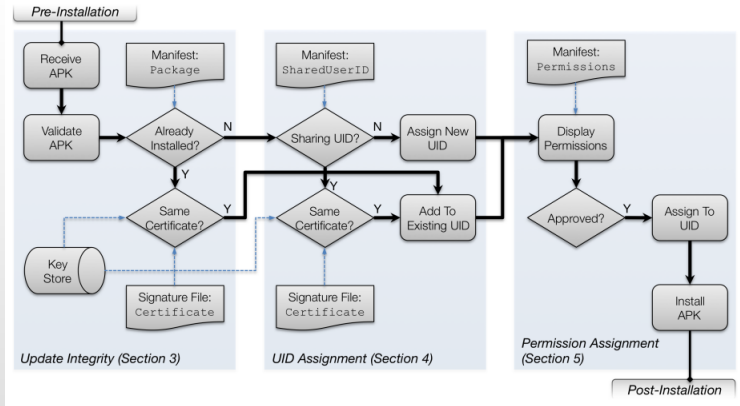
- Incorporación de compilación *Ahead-of-time* (AOT) utilizando **dex2oat** además de la compilación *just-in-time* (JIT) existente.
- Optimización del garbage collection (GC).

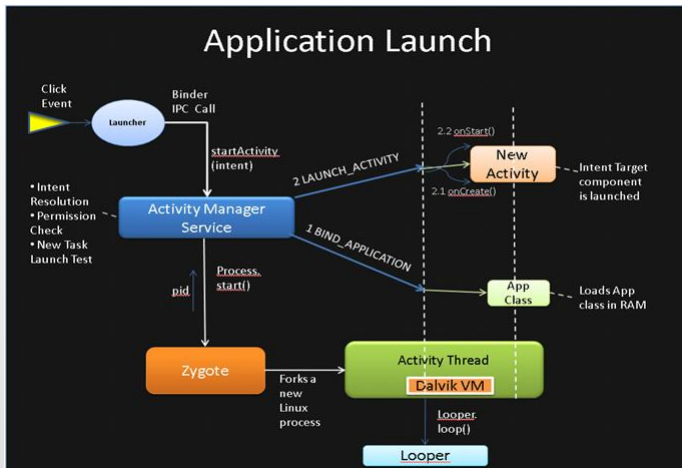
²<https://source.android.com/devices/tech/dalvik/dex-format>



- Un *android package* contiene todo lo necesario para ejecutar la aplicación en un dispositivo.
- **Gradle** es el *application builder* oficial del *SDK* de Android (*build.gradle*).
 - Debug mode → automático.
 - Release mode.







- Ninguna aplicación puede ejecutar operaciones que afecten a las demás.
- Solo pueden escribir y leer datos privados de la aplicación.
- Las aplicaciones comparten datos de manera explícita.



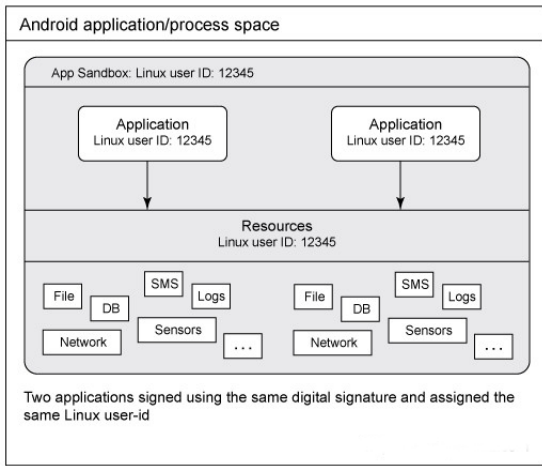
Seguridad - Id. de usuario y permisos sobre archivos

- Cuando se instala un *.apk*, Android le otorga un *userId* de Linux definitivo.
- En otro dispositivo el mismo paquete podría tener otro *userId*.
- Dos aplicaciones con el mismo *userId* son tratadas como la misma aplicación.
- Mismo *userId* = Mismo usuario Linux = Misma aplicación = Mismos permisos (*UGO*) sobre los archivos de la aplicación.
- Las aplicaciones que comparten el *userId* tienen que compartir la firma. Es decir que deben ser firmados por la misma clave privada.

```
# ls -l data/data/ | grep brow
# drwxr-x--x u0_a14 u0_a14 2016-05-01 17:55 com.
  android.browse
```



Seguridad - Ejemplo shared user id



- Se debe declarar el acceso a los recursos de manera estática (*Manifest.xml*) → dinámico a partir de v6.0
- Cuando la aplicación es instalada el usuario debe dar su consentimiento.
- *SecurityException*

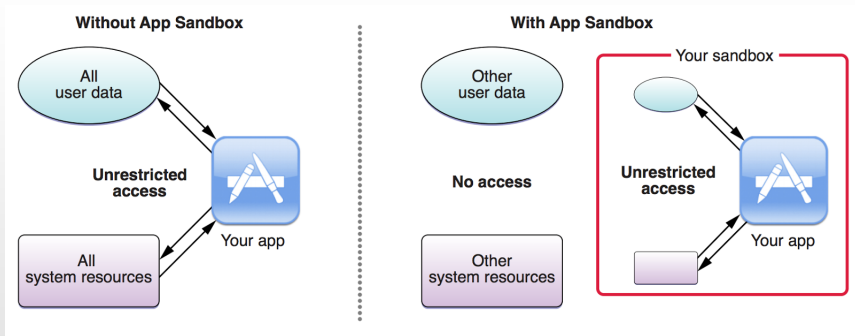
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >

    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.INTERNET" />

    ...
</manifest>
```



Seguridad - Application sandbox



- Solo una porción de código Android corre como *root*.
- ¿Cómo hacen diferentes reproductores para acceder a la música?



- Todas las aplicaciones (.apk) tienen que ser firmados digitalmente.
- Se emiten certificados auto-firmados (*KeyTool*):
 - Debug mode → *debug key* → el *keystore* se crea automáticamente (*\$HOME/.android/debug.keystore*).
 - Release mode → *developer's private key* → manualmente.



- A nivel de *AndroidManifest* → permisos a recursos que termina de conceder o configurar el usuario (aplicado por el *sandbox*).
- A nivel de las aplicaciones y sus archivos → certificado, *userId* y permisos *UGO* (aplicado por el *sandbox*).
- A nivel de file system → particiones *read-only* (/system).
- A nivel de *Linux* y *ART* (aislamiento de aplicaciones).



- Almacenamiento interno
 - *MODE_PRIVATE*, *MODE_WORLD_READABLE* o *MODE_WORLD_WRITEABLE* → desde v7.0 *FileProvider*.
- Shared preferences [ref]
 - Se puede manejar una colección de ellas.
 - Se almacenan a través de archivos XML.
 - Aplican como almacenamiento interno.
 - */data/data/<package_name>/shared_prefs/*
- Almacenamiento externo → *READ_EXTERNAL_STORAGE* y *WRITE_EXTERNAL_STORAGE*
- SQLite:
 - Bajo licencia *GPL*.
 - Actúa sobre archivos ordinarios.
 - Cumple las propiedades ACID.
 - */data/data/<package_name>/databases/*



- Raw NAND flash:
 - Subsistema *MTD* (Memory Technology Device).
 - */dev/block/mtdblockN*

```
$ cat /proc/mtd
dev:      size  erasesize  name
mtd0: 05660000 00020000 "system"
mtd1: 04000000 00020000 "userdata"
mtd2: 04000000 00020000 "cache"
```

- SD, MicroSD y eMMC (Flash Translation Layer):
 - Driver *mmcblk* (Multimedia Card Block).
 - */dev/block/mmcblk[chip number]p[partition number]*

```
$ ls -l /dev/block/platform/msm_sdcc.1/by-name
cache -> /dev/block/mmcblk0p36
system -> /dev/block/mmcblk0p38
userdata -> /dev/block/mmcblk0p40
```



- YAFFS
 - Booteo más rápido.
 - Consume menos memoria.
 - Divide los archivos en páginas.
 - *GPL*.
 - v1 y v2.
 - Single-threading.
 - Por lo general, hasta **Froyo** (v2.2).
 - Especializado para funcionar en memorias de tipo raw NAND.
- Ext4
 - Multi-threading.
 - Desde **Gingerbread** (v2.3).
 - Utilizado generalmente en memorias SD, MicroSD o eMMC.



File system - Puntos de montaje principales

- */system*
- */system/bin*
- */data/app*
- */data/data/<package_identifier>*
- */recovery*
- */boot*
- */cache*



- La idea de **Google** es mantener *GPL* fuera del espacio de usuario:
 - *Apache v2* para el espacio de usuario.
 - *GPL* para el espacio de kernel.
- Además de tamaño y optimización, esta fue otra de las razones por las que implementaron su propia versión de *libc* (*BIONIC*).



1 Overview

- Principales características
- Etimología e Historia
- Versionado
- Stack
- Procesos e Hilos
- DVM/ART vs. JVM
- Aplicaciones
- Seguridad
- Almacenamiento
- File System
- Licencia

2 Rooting

- Definición
- Bootloader
- Boot.img & initramfs
- Booteo/flasheo y testeo

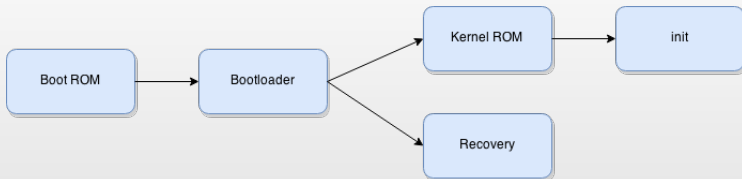


- Rooting Android device → darle permisos de *root* al dispositivo.
- Jailbreaking iOS device → temabién necesario para instalar aplicaciones no autorizadas.



Bootloader - Definición [ref]

- Inicializa el hardware y carga el *initramfs*.
- Permite bootear una imagen → Android o recovery.
- Esta fuera de lo que es Android.



- ¿Por qué esta bloqueado si Android es *open source*?
- Desbloquearlo implica perder la garantía del dispositivo → impuesto por *Digital Rights Management*.



- Protocolo USB.
- Permite bootear customs ROMs.
- Permite flashear una partición.

```
fastboot flash boot out/target/product/hikey/boot.  
img
```

- Permite desbloquear el bootloader (solo si el dispositivo lo soporta).

```
fastboot oem get_unlock_data
```

```
fastboot oem unlock [ unlock code ]
```



- Open bootloader → las compañías no lo desean.
- Locked bootloader *Carrier ID* → pérdida de garantía y reseteo de fábrica.



Boot.img & initramfs - Boot.img

- Obtención:
 - Extracción de una imagen original → de un dispositivo o del SDK de Android.
 - Descarga desde un sitio confiable (MIUI/LineageOS) → *zip*.
- División:
 - A través de *unmkbootimg*.
 - Estructura:

```
+-----+
| boot header | 1 page
+-----+
| kernel      | n pages
+-----+
| ramdisk     | m pages
+-----+
| second stage| o pages
+-----+
```

```
n = (kernel_size + page_size - 1) / page_size
m = (ramdisk_size + page_size - 1) / page_size
o = (second_size + page_size - 1) / page_size
```

- 0. all entities are *page_size* aligned in flash
- 1. kernel and ramdisk are required (*size != 0*)
- 2. second is optional (*second_size == 0* -> no second)



- Desempaquetado y descompresión de initramfs (*cpio* + *gzip*):

```
gunzip -c ../your-ramdisk-file | cpio -i
```

- Archivo *default.prop* → contiene propiedades de configuración para la inicialización del sistema.
- Habilitar modo inseguro: propiedad *ro.secure* debe estar en falso → demonio de adb (*adbd*) con permisos de *root*.
- Empaquetar y comprimir initramfs inseguro:

```
find . | cpio -o -H newc | gzip > ../newramdisk.cpio.gz
```



- A través de *mkbootimg*.

```
mkbootimg --kernel zImage --ramdisk  
insecure_initramfs.cpio.gz --base 0x80200000 --  
cmdline 'androidboot.hardware=qcom user_debug=31  
zcache' -o new_boot.img
```



- Booteo (método volátil):

```
$ fastboot boot new_boot.img
```

- Flasheo (método no volátil):

```
$ fastboot flash boot new_boot.img
```

- Acceso al dispositivo mediante *adb* como usuario *root* en lugar de shell.

```
$ adb shell
```



¿Preguntas?

