



Procesos en Android

LIN - Curso 2015-2016





Contenido

1 Introducción

2 Componentes de una aplicación Android

3 Construcción, distribución y ejecución de aplicaciones

- Lanzamiento de aplicaciones
- Android Debug Bridge

4 Tutorial: Instalación de aplicación en MV de Android-x86



Contenido

1 Introducción

2 Componentes de una aplicación Android

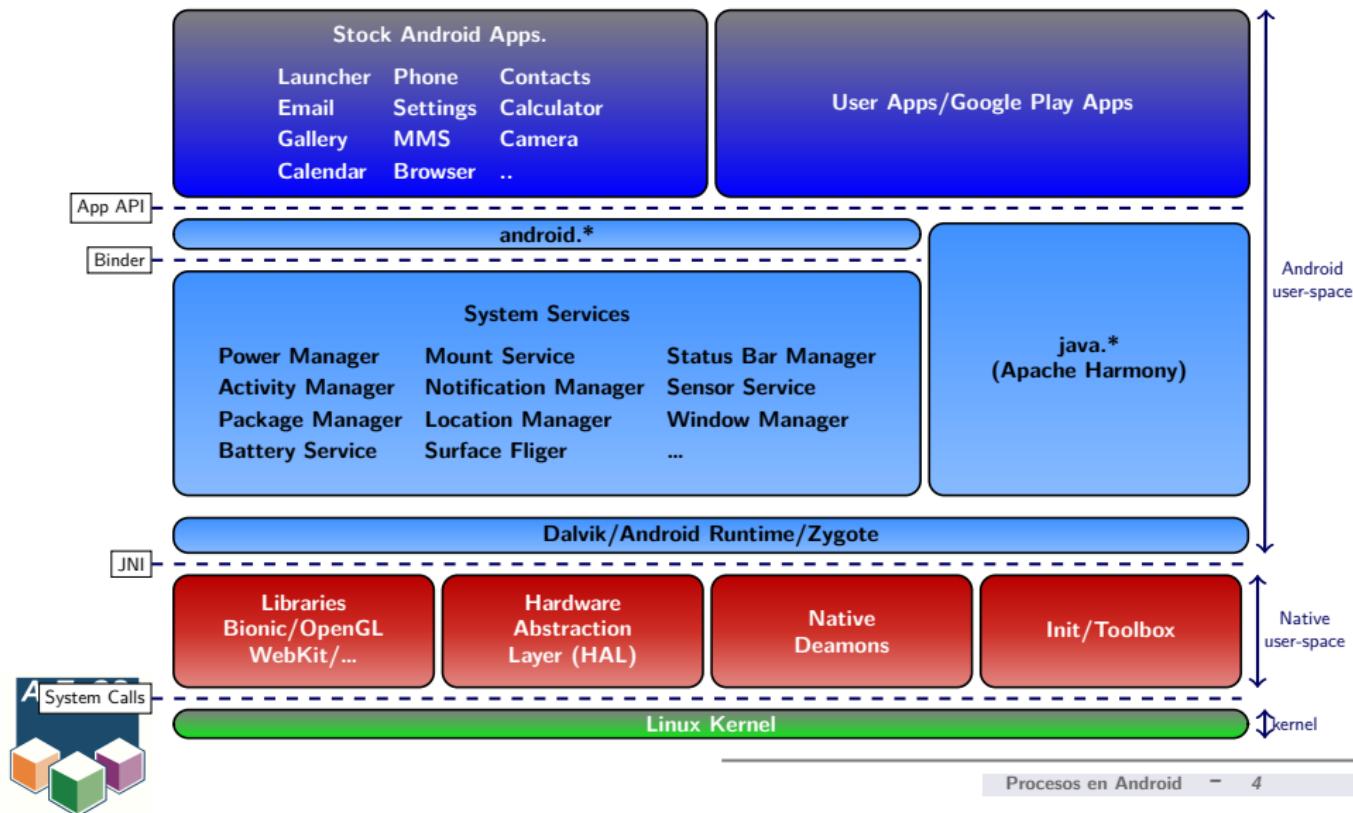
3 Construcción, distribución y ejecución de aplicaciones

- Lanzamiento de aplicaciones
- Android Debug Bridge

4 Tutorial: Instalación de aplicación en MV de Android-x86



Arquitectura de Android





Introducción

Dos tipos de aplicaciones en Android:

- 1 Aplicaciones Android/Java (*Android user-space*)
- 2 Aplicaciones/Demonios nativos (*Native user-space*)
 - Similar a aplicaciones nativas en Linux
 - Ejemplo: comandos lanzados con *ADB shell*

Enfoque

- En este tema nos centraremos en las aplicaciones Android (Java)
- El desarrollo de aplicaciones Android está fuera del temario de LIN
 - Aspecto ampliamente documentado
 - <http://developer.android.com/develop/index.html>

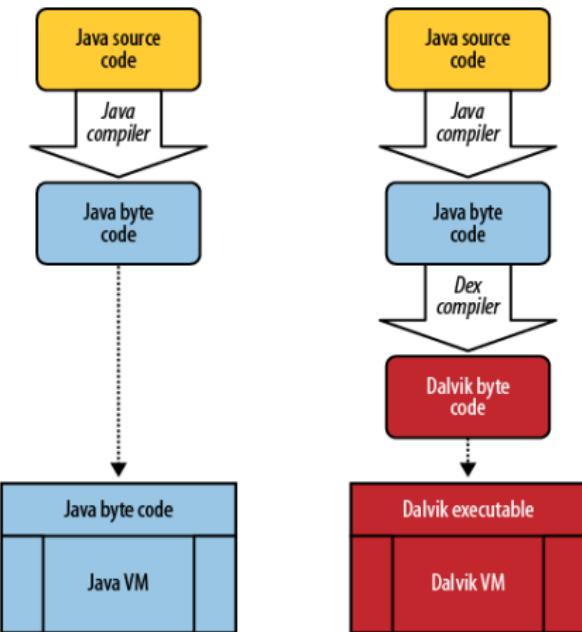


Aplicaciones en Android (I)

- Aplicaciones Android están escritas en Java pero **usan Dalvik VM (o ART)** en lugar de Java VM

Dalvik/ART vs. Java VM

- Basada en registros/Basada en pila
- Ejecución de ficheros .dex vs. .class
- Dalvik/ART poseen optimizaciones para móviles





Aplicaciones en Android (II)

Cada aplicación ...

- Se ejecuta en un **proceso Linux separado** (Instancia de MV Dalvik/ART)
 - Puede constar de uno o varios hilos
 - Excepcionalmente distintos componentes de la aplicación se podrían asignar a distintas MVs
 - Múltiples procesos se comunican vía Binder
- Tiene asociado **un usuario único (UID)**
 - El usuario se crea al instalar la aplicación
 - No confundir con los “usuarios humanos” del sistema
- Tiene **su propio conjunto de ficheros, preferencias y base de datos “privados” (sandbox)**
 - La compartición de datos entre aplicaciones se ha de solicitar explícitamente





Aplicaciones en Android (III)

Requisitos de memoria

- Android está diseñado para funcionar en **plataformas con pocos recursos de memoria**
 - Memoria principal reducida
 - Sin swap
- Cuando el sistema se queda sin memoria, se activa el LMK (*Low Memory Killer*)
 - Mata procesos en base a ciertas prioridades
- El desarrollador debe tener en cuenta que **su aplicación puede ser finalizada** total o parcialmente (componentes)
 - Notificaciones de cambio de estado a las aplicaciones





Contenido

1 Introducción

2 Componentes de una aplicación Android

3 Construcción, distribución y ejecución de aplicaciones

- Lanzamiento de aplicaciones
- Android Debug Bridge

4 Tutorial: Instalación de aplicación en MV de Android-x86



Componentes de una aplicación Android

Una aplicación consta de 4 tipos de componentes desacoplados

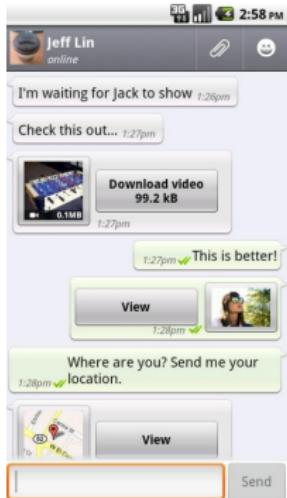
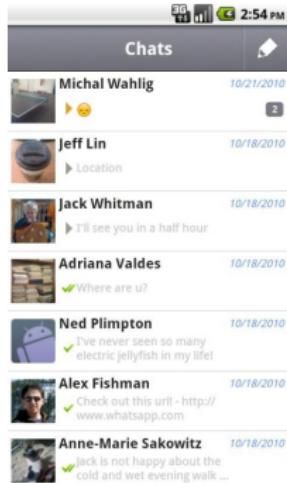
- 1 Activity**
- 2 Service**
- 3 Broadcast receiver**
- 4 Content provider**

- Cada componente es un posible punto de entrada a la aplicación
 - No hay `main()`
- La aplicación se “crea” al activarse alguno de sus componentes
 - Android emplea *Intents* para activar componentes



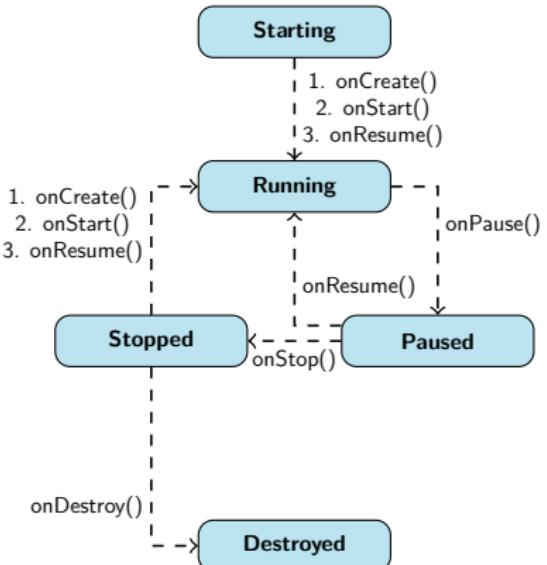
Activities

- **Activity:** Cada pantalla/ventana de la aplicación
 - GUI de la aplicación consta de una o varias *activities*
 - Cada activity se implementa usando XML y Java





Activity Life Cycle



Estados de una *activity*

- *Running* (visible totalmente)
- *Paused* (visible parcialmente)
- *Stopped* (no visible)

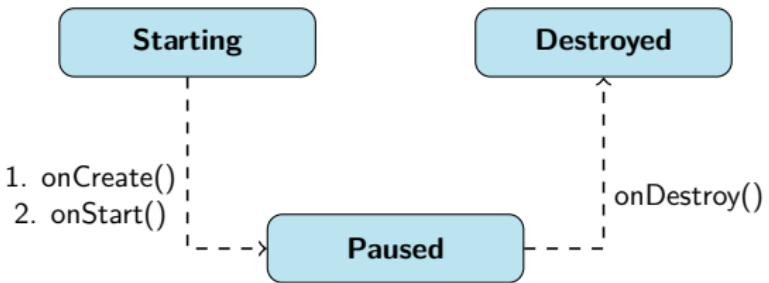
Aplicación recibe notificaciones

- Cualquier Activity *Stopped* o *Paused* podría destruirse
- Aconsejable guardar estado en transición *Running* → *Paused*



Services

- **Service:** Componente que se ejecuta en segundo plano
 - Para hacer tareas potencialmente costosas o de tiempo indefinido
 - Ej: descargar ficheros, reproducir música
 - No tiene interfaz gráfica
 - Por defecto el código de un servicio se ejecuta en el *UI thread*
 - Recomendable ejecutarlo en un *thread* separado





Broadcast Receivers y Content Providers

- **Broadcast receiver:** Recibe mensajes globales del sistema y reacciona en consecuencia
 - Eventos: Batería baja, 3G/4G off, Wifi on
 - Acciones típicas: dejar de actualizar datos de Internet, mostrar publicidad, ...
 - Patrón de diseño *Observer*
- **Content provider:** Permite que los datos de la aplicación estén disponibles para otros componentes de la misma u otras aplicaciones
 - Interfaz CRUD
 - Ejemplo: Content Provider para los *Contactos*



Intents

- **Intent:** mensaje que una aplicación envía al sistema para activar un componente de la misma u otra aplicación
 - Abstracción clave de Android
- Dos tipos de *intent*:
 - **Explícito:** Activar el componente "X" de la aplicación "Y"
 - Ejemplo: Lanzar una aplicación → activar *main Activity*
 - **Implícito:** Activa un componente del sistema capaz de hacer una tarea concreta
 - Ejemplo: Abrir aplicación capaz de leer un archivo PDF o página web
 - Se declaran en *manifest* de la aplicación

Completar la acción mediante

 Adobe Reader	 Kingsoft Office
 Lector de PDF de Drive	
Siempre	Solo una vez



Contenido

1 Introducción

2 Componentes de una aplicación Android

3 Construcción, distribución y ejecución de aplicaciones

- Lanzamiento de aplicaciones
- Android Debug Bridge

4 Tutorial: Instalación de aplicación en MV de Android-x86



Android APKs (I)

- Las aplicaciones se empaquetan y distribuyen en APKs
 - Archivo JAR (ZIP) con estructura predefinida
 - Más información estructura APK [aquí](#)
 - Se pueden alterar e inspeccionar con aapt (SDK)





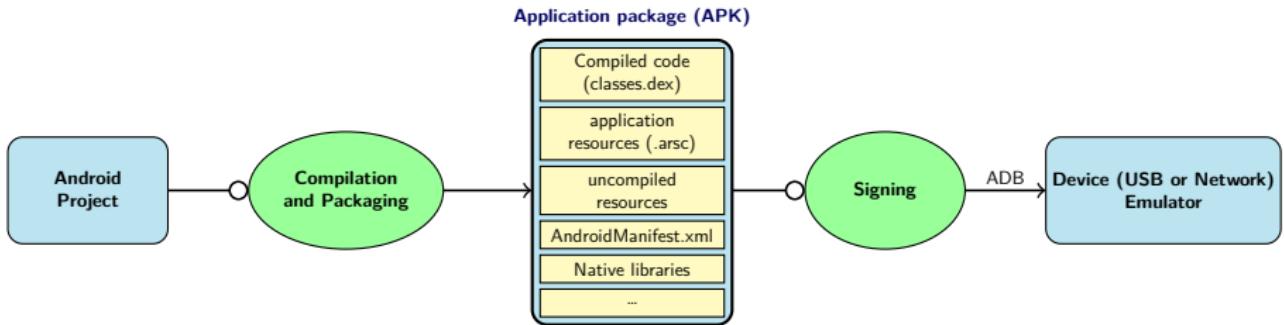
Android APKs (II)

Tres formas de distribución de APK

- Obtener aplicación a través de un *market*
 - Google Play
 - F-Droid
 - Amazon Appstore
 - ...
- Descargar APK de un sitio web
- Transferir APK desde un host al dispositivo vía ADB



Construcción e Instalación de Aplicaciones



Herramientas de desarrollo

- Android Studio
- Android SDK
- Android NDK



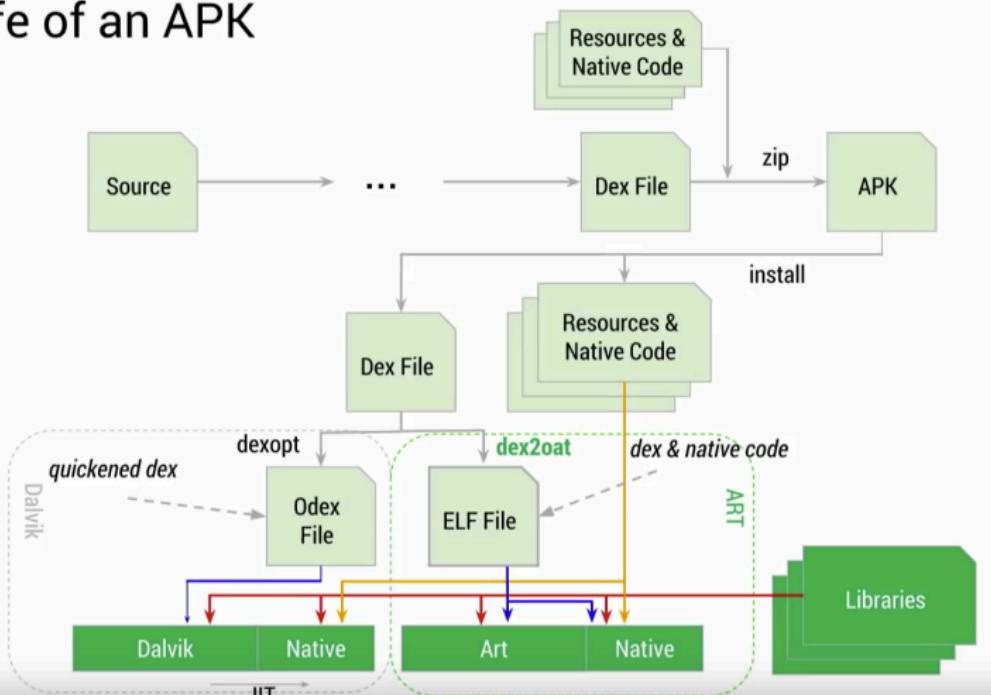
Instalación de aplicaciones

- Por motivos de rendimiento el código dex de un fichero APK no se ejecuta directamente en MV Dalvik/ART
 - Android realiza optimizaciones en tiempo de instalación
- Android genera ficheros intermedios (ODEX,OAT,...) preprocesando ficheros Jar/Dex
 - Ficheros generados se almacenan en “/data/dalvik-cache”
- Para una aplicación los ficheros se generan:
 - Al arrancar por primera vez el dispositivo (*Stock apps*)
 - Tras realizar actualización del sistema
 - ... o al instalar la aplicación por primera vez (*3rd party apps*)



Instalación de aplicaciones: Dalvik vs. ART

The life of an APK



Fuente: B. Carlstrom, A. Ghuloum, I. Rogers. *The ART runtime*. Google I/O, 2014.





Instalación de aplicaciones: Dalvik vs. ART

Dalvik: MV por defecto versiones de Android \leq 4.4 (KitKat)

- Ficheros Jar/Dex se convierten a *ODEX* con dexopt
- ODEX: *Optimized (Preprocessed) DEX*
- Compilación JIT (*Just-in-time*)

ART: MV por defecto versiones de Android \geq 5.0 (Lollipop)

- Ficheros Jar/Dex se convierten a *OAT* con dex2oat
- OAT: código Dex + código nativo
 - Tipo particular de fichero ELF
- Compilación AOT (*Ahead-of-time*)
 - No todo el código Dex se traduce a código nativo



Lanzamiento de aplicaciones

- Mecanismo tradicional de creación de aplicación Java:

- 1 fork()
- 2 exec(Dalvik, class, args)
- 3 cargar clases/librerías
- 4 ejecutar primera instrucción del código

- Mecanismo tradicional no es lo más eficiente para Java

- Código clases Java no se comparte de forma natural entre procesos

- Aplicaciones Android usan **librerías comunes**:

- Core Libraries (API de Java)
 - Android Framework (API de Android)

- **Enormes oportunidades de optimización**

- Memoria
 - Tiempo de creación



Lanzamiento de aplicaciones: Zygote

- **Zygote:** Proceso encargado de crear aplicaciones Java mediante clonación
 - Cigoto: *Célula inicial formada durante la creación de un organismo*

Características

- Proceso creado por init
- Ejecuta instancia de Dalvik/ART
- Tiene librerías Java precargadas



Lanzamiento de aplicaciones: Zygote (cont.)

Creación de aplicaciones vía Zygote

- Zygote está a la espera de peticiones de creación de aplicaciones escuchando en un socket
- Al recibir petición de creación crea un nuevo proceso vía fork()
 - Proceso hijo ejecuta Dalvik/ART pero con librerías precargadas



Lanzamiento de aplicaciones: Zygote (cont.)

Creación de aplicaciones vía Zygote

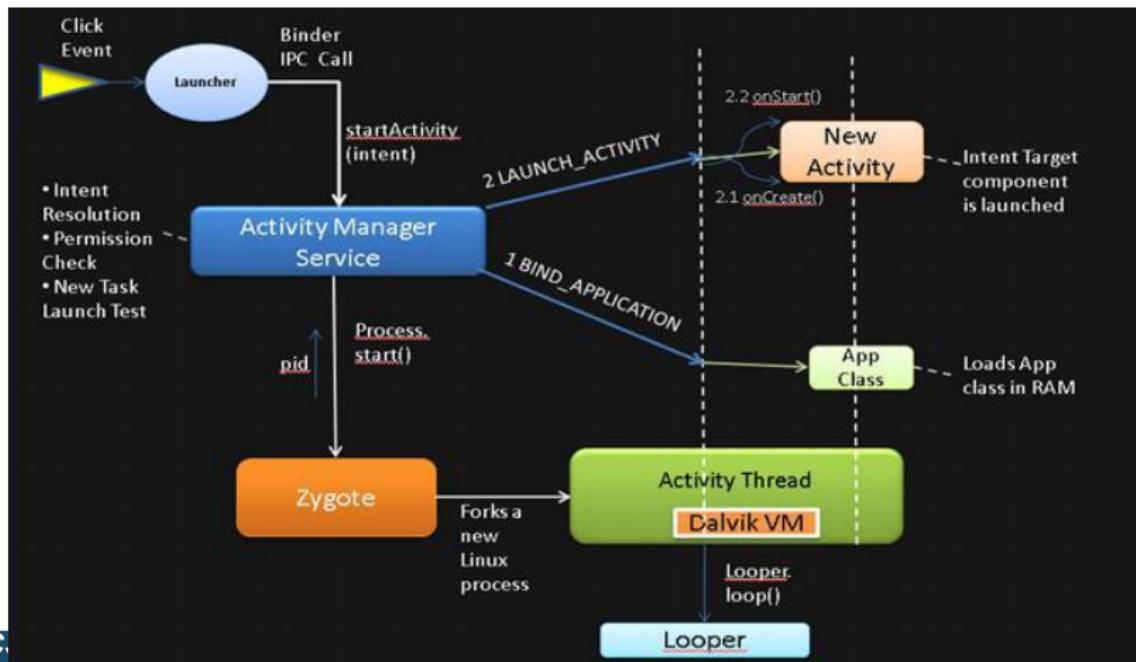
- Zygote está a la espera de peticiones de creación de aplicaciones escuchando en un socket
- Al recibir petición de creación crea un nuevo proceso vía fork()
 - Proceso hijo ejecuta Dalvik/ART pero con librerías precargadas

Linux usa fork() con COW → todo son ventajas

- 1 Compartición código de librerías Java entre TODAS las aplicaciones
- 2 Sólo se duplican bajo demanda las páginas modificadas
 - Ejemplo: modificación de atributo static de una clase



Lanzamiento de aplicaciones: Visión general





Android Debug Bridge (ADB)

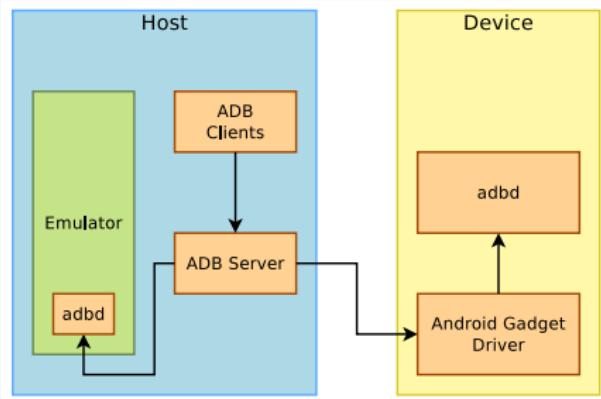
- ADB: Herramienta creada por Google para depuración sobre dispositivos con Android
 - Soporta conexión por USB o red al dispositivo
- Otras funciones de ADB:
 - Iniciar un shell interactivo
 - Transferir ficheros entre el host y el dispositivo
 - Instalar/desinstalar aplicaciones
 - ...



Android Debug Bridge (ADB)

Componentes de ADB

- 1 **ADBd**: demonio que se ejecuta en el dispositivo o en el emulador
- 2 **ADB server**: servidor que se ejecuta en el host de depuración
- 3 **Programa adb**: cliente que permite enviar comandos al dispositivo o al emulador





Algunos comandos básicos de 'adb'

Comando	Descripción
help	Listado de comandos disponibles
start-server	Arranca ADB server en el host
kill-server	Mata ADB server en el host
devices	Lista dispositivos Android detectados
connect	Conecta ADB server con ADBd de un dispositivo por red
disconnect	Desconectar conexión iniciada con dispositivo remoto
shell	Inicia un shell interactivo en el dispositivo. También es posible ejecutar un único comando del shell con <code>adb shell <comando></code>
root	Reinicia ADBd con permisos de root Válido para dispositivos “rooteados”
push	Copia fichero local al dispositivo Sintaxis: <code>adb push <origen> <destino></code>
pull	Copia fichero de dispositivo al host Sintaxis: <code>adb pull <origen> <destino></code>

¹Uso: `adb <comando> [opciones]`. Es posible encontrar más información aquí.



Contenido

-
- 1 Introducción**
 - 2 Componentes de una aplicación Android**
 - 3 Construcción, distribución y ejecución de aplicaciones**
 - Lanzamiento de aplicaciones
 - Android Debug Bridge
 - 4 Tutorial: Instalación de aplicación en MV de Android-x86**



Tutorial: Instalación de aplicación en MV de Android-x86

- Se proporciona fichero .tgz que contiene los siguientes componentes:
 - Software de desarrollo
 - 1 Android Studio
 - 2 Android SDK
 - 3 Android NDK
 - Proyecto de ejemplo “Hello”
- Ubicación del fichero (accesible desde “usuario_vms”):
 - /mnt/DiscoVMs/LINyAISO/Android-devtools-LIN.tgz
 - También puede descargarse de Google Drive ([link](#))
- Alternativamente, el software de desarrollo puede obtenerse en sitio oficial:
 - <http://developer.android.com/sdk/index.html>
 - <https://developer.android.com/tools/sdk/ndk/index.html>



Tutorial: Instalación de aplicación en MV de Android-x86

Pasos

- 1** Instalación de Software de Desarrollo de Android
- 2** Configuración inicial de Android Studio
- 3** Abrir proyecto “Hello” en Android Studio
- 4** Conectar MV de Android-X86 con Android Studio vía ADB
 - En este punto es preciso arrancar la MV
- 5** Instalar y ejecutar aplicación “Hello” en MV de Android-X86



Instalación del Software de Desarrollo

- Iniciar sesión con `usuario_vms` y abrir una terminal
- Eliminar ficheros de configuraciones anteriores

```
$ rm -rf ~/.android  
$ rm -rf ~/.AndroidStudio
```

- Localizar fichero `Android-devtools-LIN.tgz`
- Descomprimir fichero `tgz` en `$HOME`

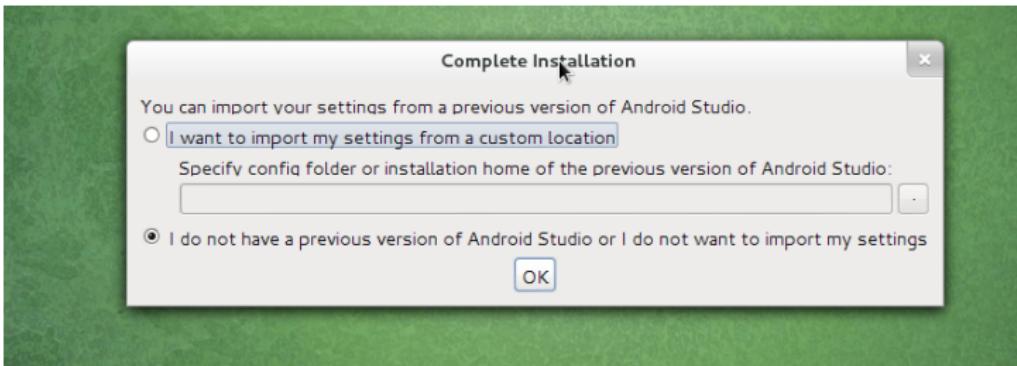
```
$ cd  
$ tar xzvf /mnt/DiscoVMs/LINyAISO/Android-devtools-LIN.tgz
```

- Ejecutar Android Studio

```
$ cd Android-dev/android-studio/bin  
$ ./studio.sh
```



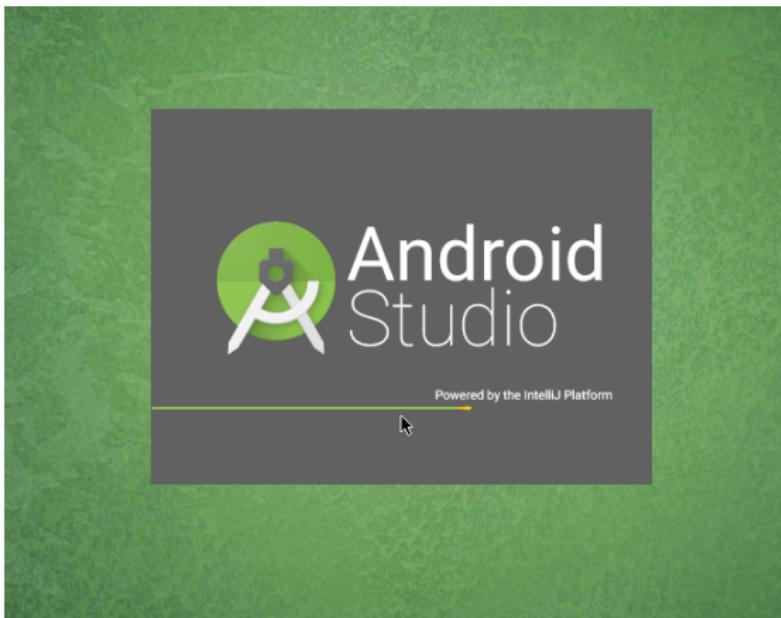
Configurando Android Studio (1/12)



- 1 Seleccionar no importar nada
- 2 Click "OK"

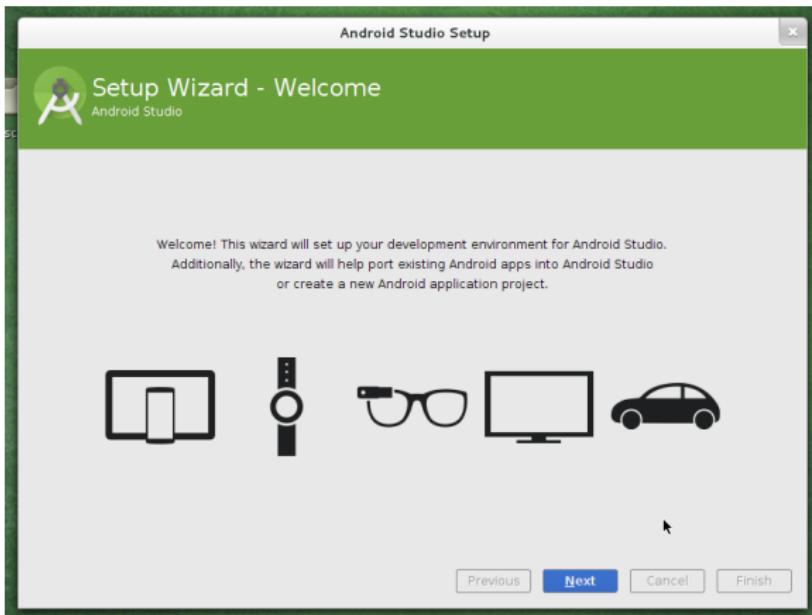


Configurando Android Studio (2/12)





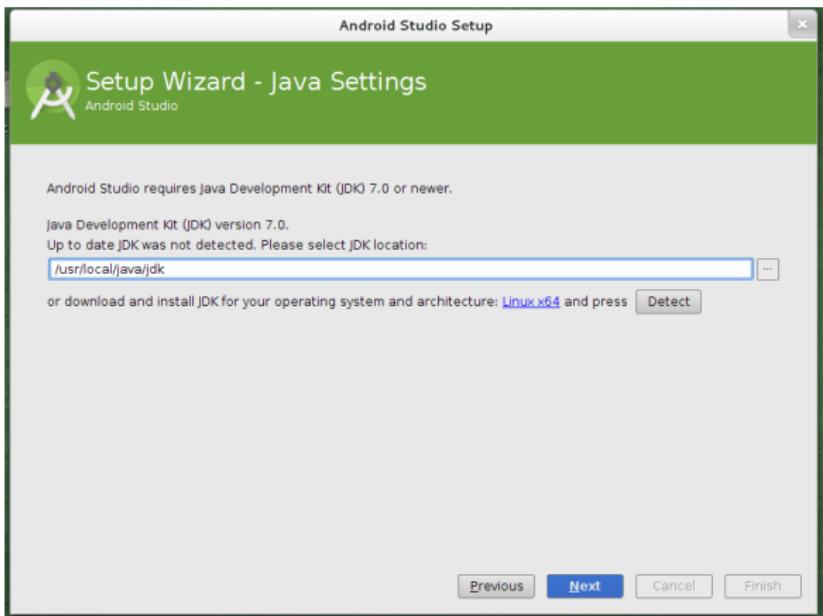
Configurando Android Studio (3/12)



Click “Next”



Configurando Android Studio (4/12)



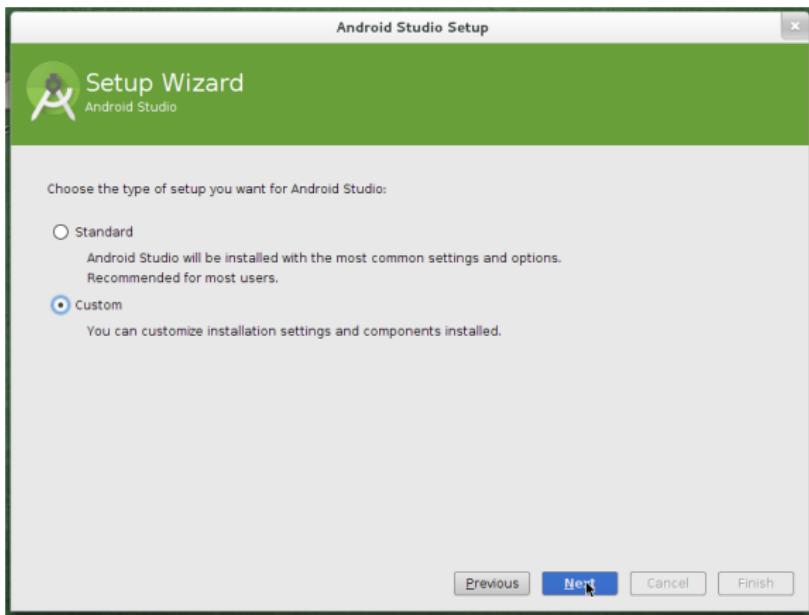
1 Seleccionar directorio donde está instalado el JDK de Oracle

- /usr/local/java/jdk

Click "Next"



Configurando Android Studio (5/12)



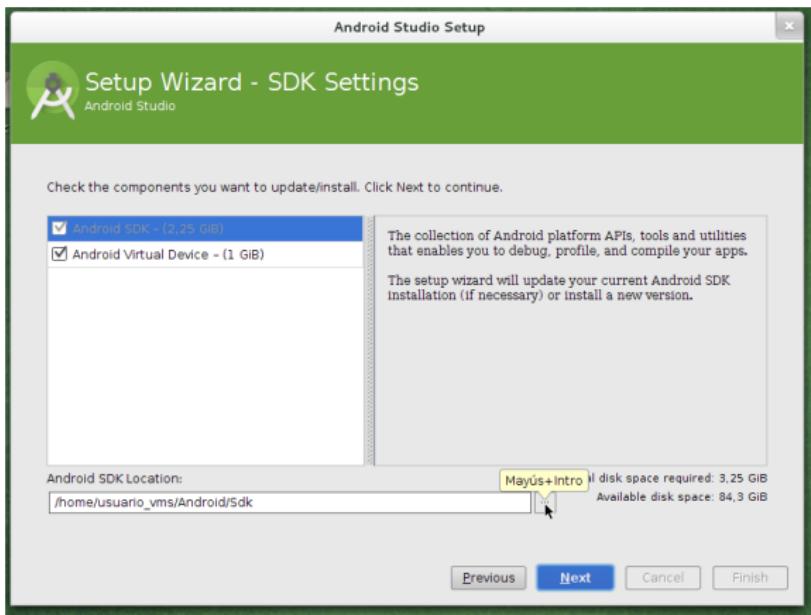
1 Seleccionar instalación “Custom”

2 Click “Next”





Configurando Android Studio (6/12)

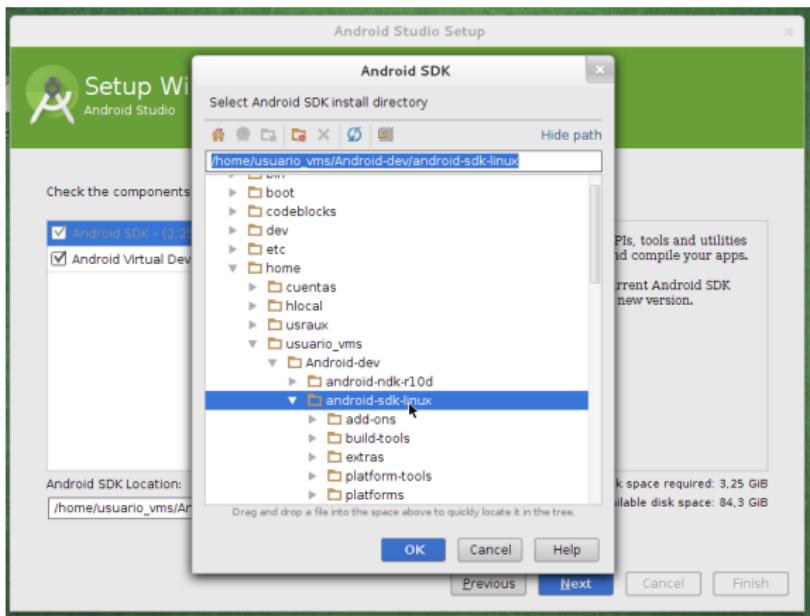


Seleccionar el directorio donde está instalado Android SDK

- Ruta: ~/Android-dev/android-sdk-linux

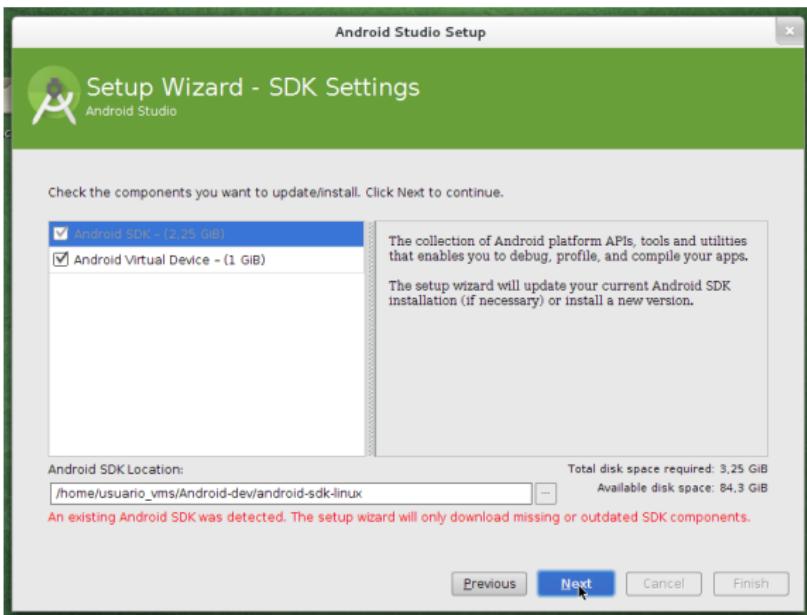


Configurando Android Studio (7/12)





Configurando Android Studio (8/12)

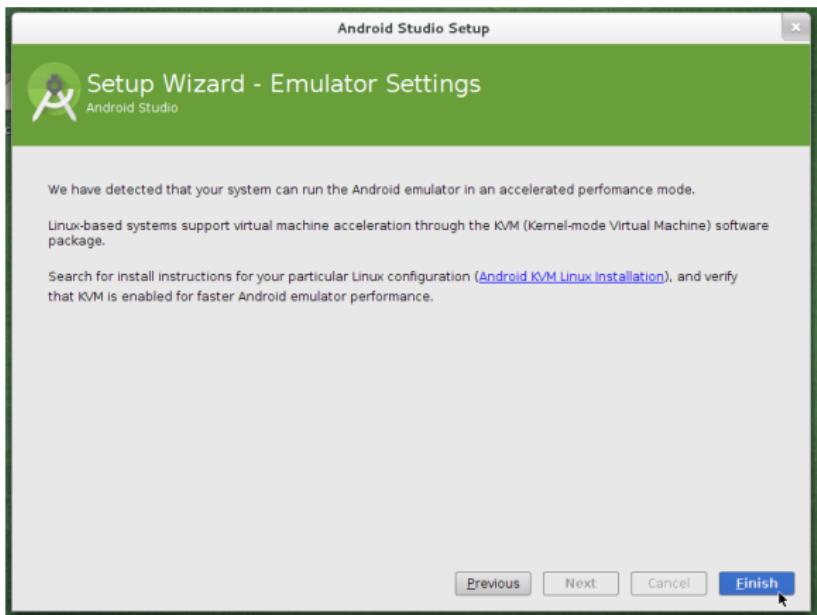


1 Se detecta SDK válido en ese directorio
2 Click "Next"





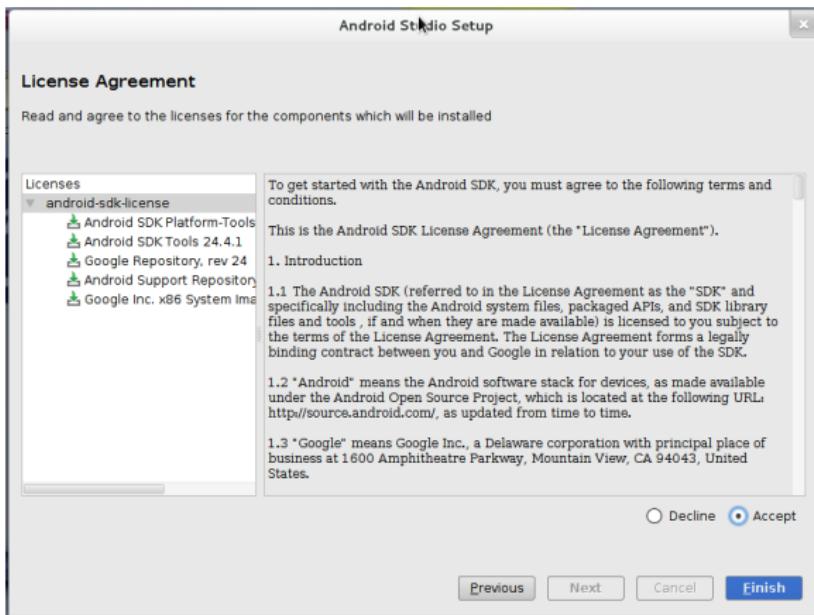
Configurando Android Studio (9/12)



Click “Finish”



Configurando Android Studio (10/12)



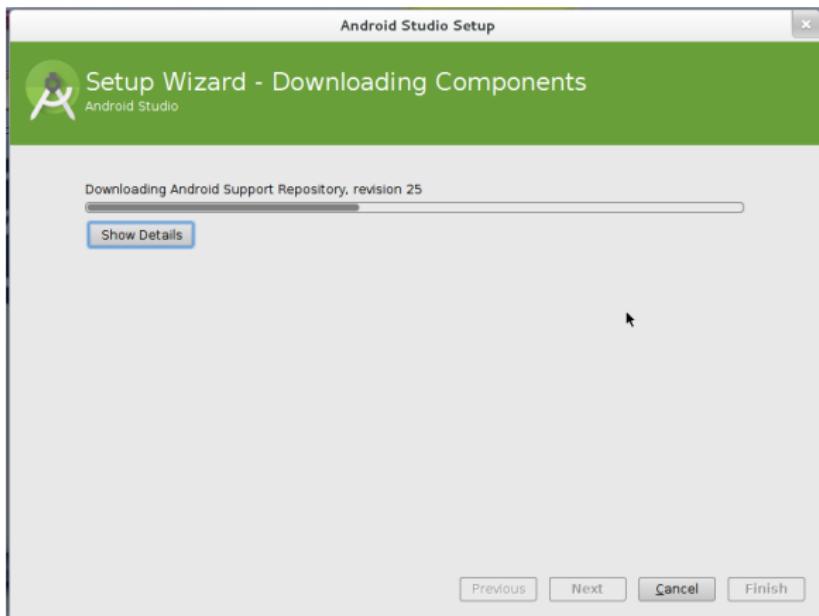
Actualizar SDK

- 1 Aceptar licencia
- 2 Click "Finish"





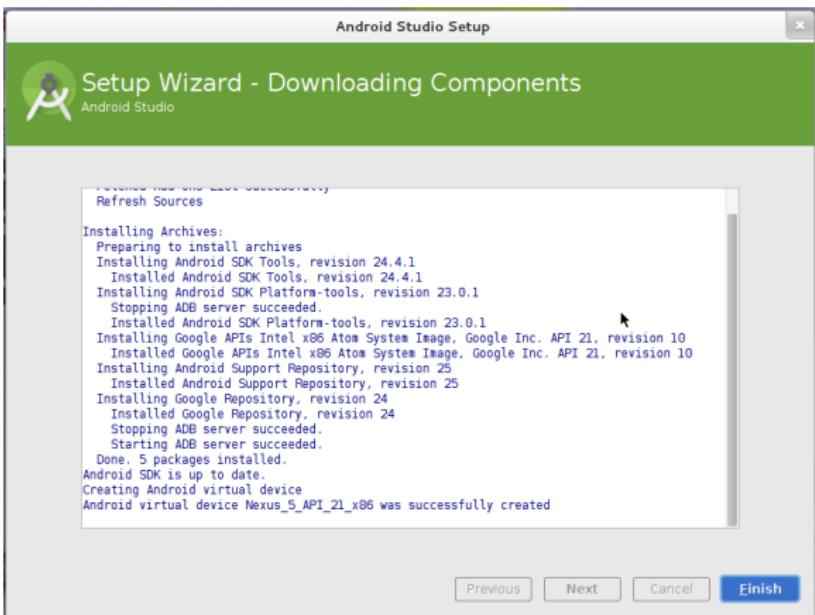
Configurando Android Studio (11/12)



Esperar a que finalice la descarga



Configurando Android Studio (12/12)



*Esperar a que finalice el proceso de configuración
Click “Finish”*



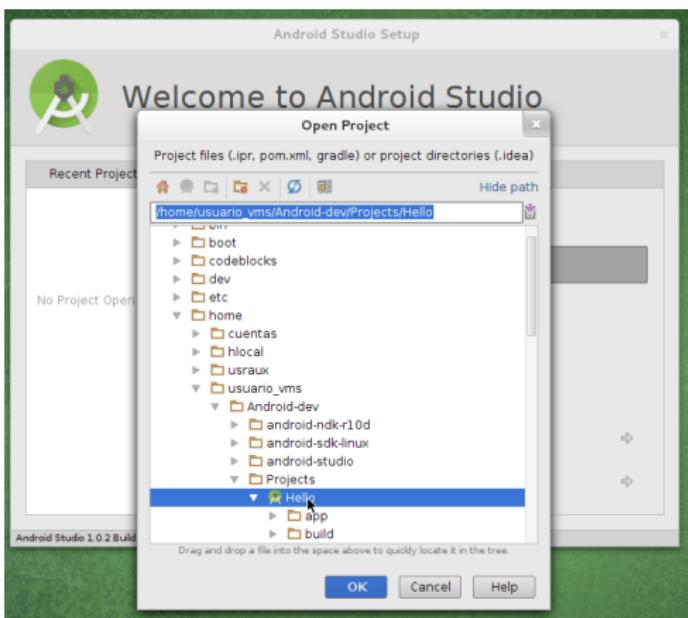
Abrir proyecto “Hello” (1/6)



Click “Open an existing Android Studio Project”



Abrir proyecto “Hello” (2/6)



1 Seleccionar directorio del proyecto

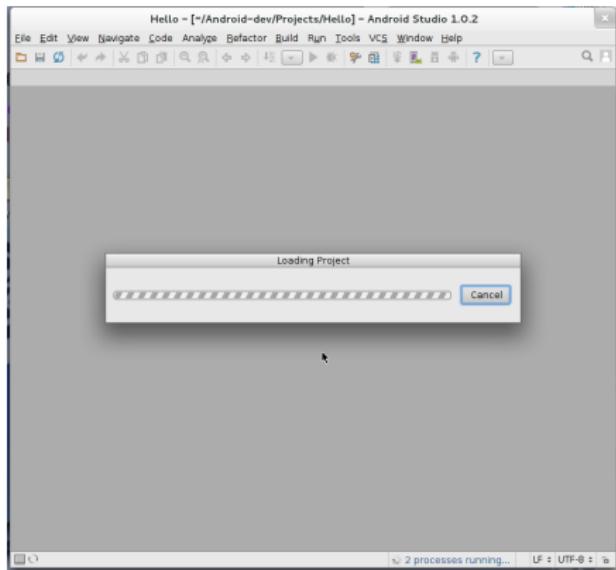
- ~/Android-dev/Projects/Hello

2 Click “OK”



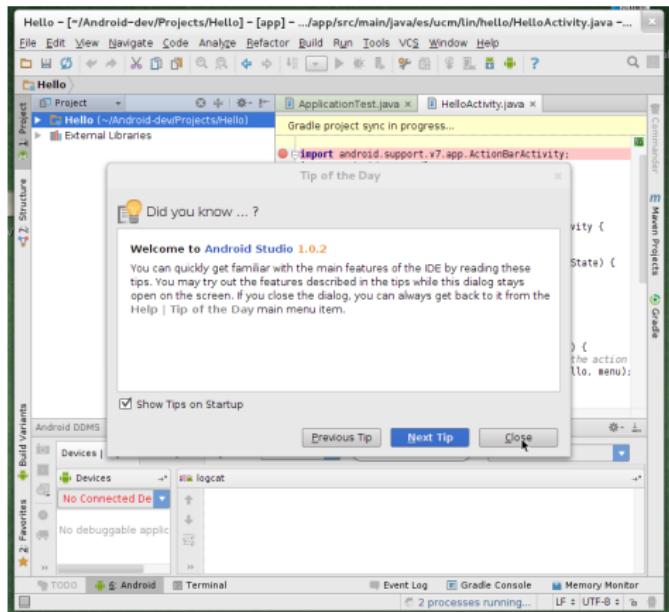


Abrir proyecto “Hello” (3/6)





Abrir proyecto “Hello” (4/6)



Click “Close”



Abrir proyecto “Hello” (5/6)

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class HelloActivity extends ActionBarActivity {

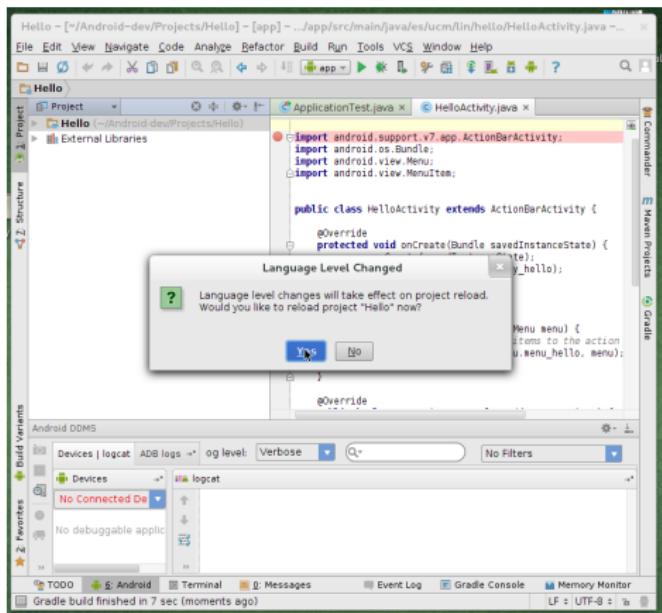
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action
        getMenuInflater().inflate(R.menu.menu_hello, menu);
        return true;
    }
}
```

Esperar a que se actualice el proyecto Gradle ...



Abrir proyecto “Hello” (6/6)



Click “Yes”



Ejecutar aplicación “Hello” en MV (1/9)

The screenshot shows the Android Studio interface with the 'Hello' project open. The code editor displays `HelloActivity.java` which extends `ActionBarActivity`. The `onCreate` method is overridden to set the content view to `R.layout.activity_hello`. The `onCreateOptionsMenu` method inflates the menu from `R.menu.menu_hello`. The bottom panel shows the Logcat tab with the message 'No Connected Devices'.

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class HelloActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_hello, menu);
        return true;
    }

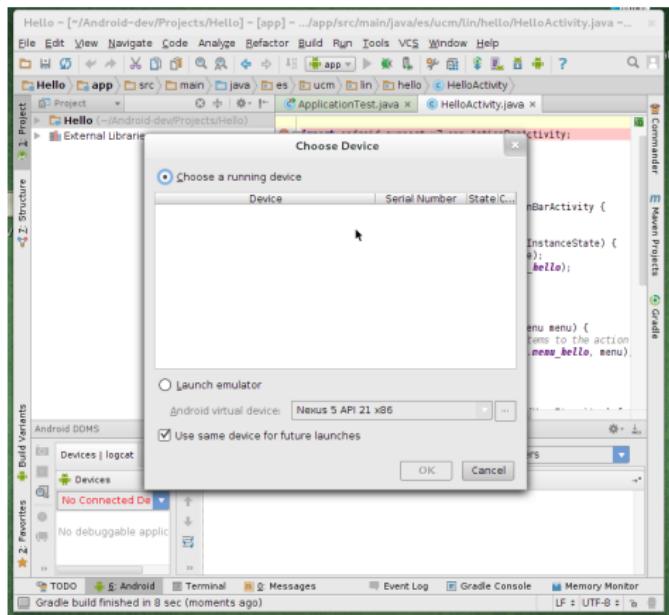
    @Override
}
```

Ejecutar aplicación dándole al “Play” ...

- La aplicación se compila automáticamente



Ejecutar aplicación “Hello” en MV (2/9)



■ Seleccionar el modo “Choose a running device”

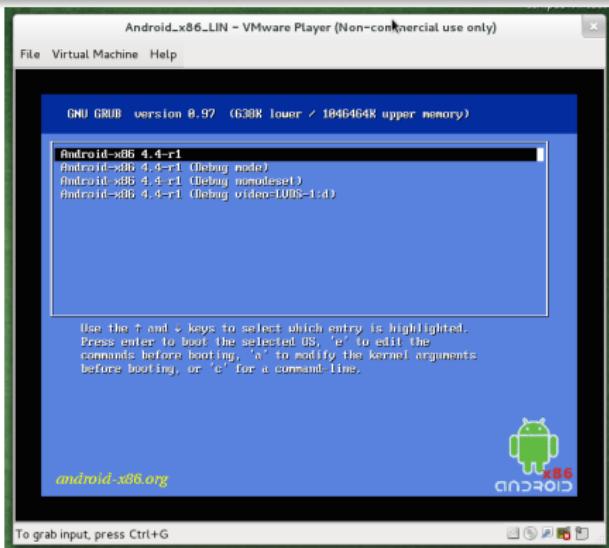
- ¡No hay ninguno!
- Es el momento de arrancar la MV



Ejecutar aplicación “Hello” en MV (3/9)

Arrancar máquina virtual de Android

```
$ VM_LINyAISO-android.sh
```



Seleccionar primera entrada en el gestor de arranque





Ejecutar aplicación “Hello” en MV (4/9)

Conectarse a la máquina virtual vía ADB

- 1 Ir al directorio donde está instalado ADB del SDK

```
$ cd ~/Android-dev/android-sdk-linux/platform-tools
```

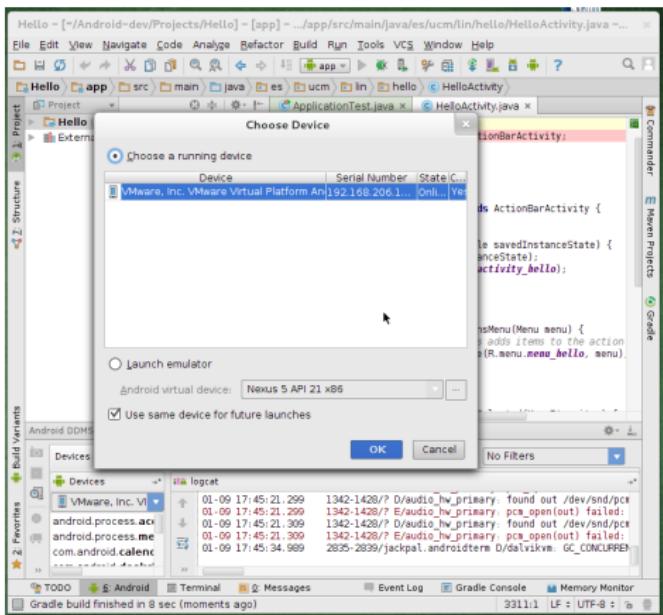
- 2 Conectar MV con ADB por red

- Sintaxis: ./adb connect [dirección IP]
 - Es posible obtener la dirección IP ejecutando el comando netcfg en la aplicación de Terminal dentro de la MV
- Conexión

```
$ ./adb connect 192.168.206.134
```



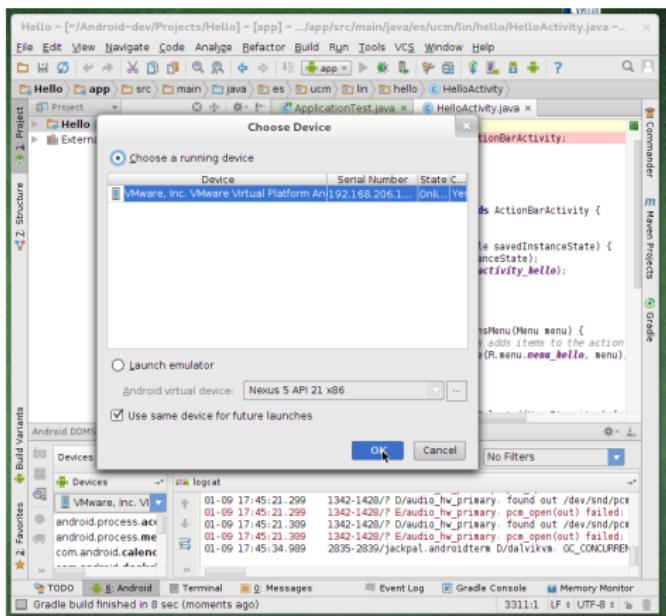
Ejecutar aplicación “Hello” en MV (5/9)



Ahora muestra un dispositivo conectado :-)



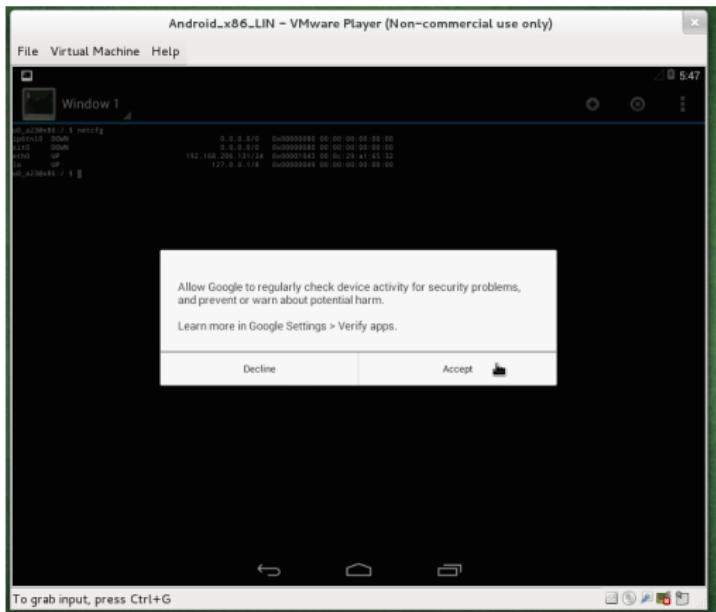
Ejecutar aplicación “Hello” en MV (6/9)



Click “OK”



Ejecutar aplicación “Hello” en MV (7/9)



1 Cambiar a la MV
2 Click “Accept”





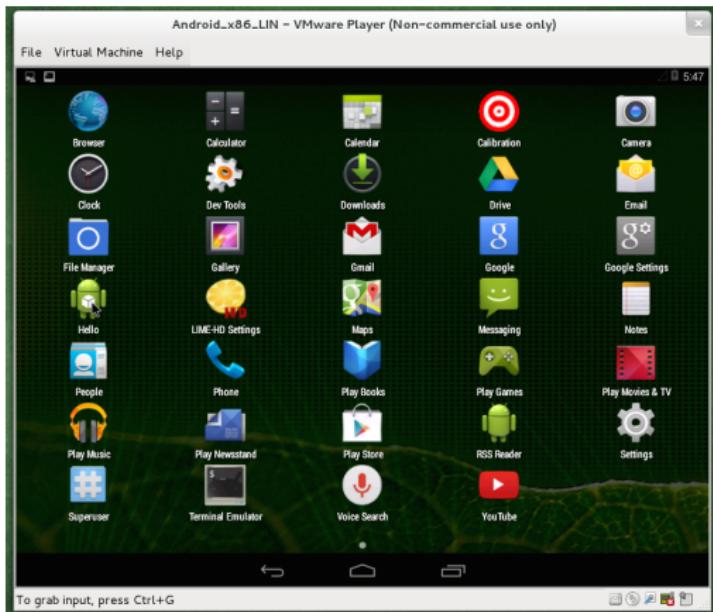
Ejecutar aplicación “Hello” en MV (8/9)



¡¡ Nuestra aplicación se está ejecutando !!



Ejecutar aplicación “Hello” en MV (9/9)



La aplicación se ha instalado y aparece en el Android Launcher



Referencias (I)

- Learning Android
 - Cap. 4 "*Main Building Blocks*"
- Embedded Android
 - Cap. 2 "*Internals Primer*"
- Presentaciones
 - *Android System Development*, Free Electrons
 - *Android Internals*, Marko Magenta (Marakana)
 - [The ART runtime](#) (Google I/O 2014)



Referencias (II)

Enlaces

- ADB: <http://developer.android.com/tools/help/adb.html>
- Desarrollo de aplicaciones:
 - <https://developer.android.com/tools/building/index.html>
 - <https://developer.android.com/tools/workflow/index.html>
 - <https://developer.android.com/tools/projects/index.html>
 - <https://developer.android.com/tools/studio/index.html>
- Procesos en Android
 - <http://multi-core-dump.blogspot.com.es/2010/04/android-application-launch.html>
 - <http://coltf.blogspot.com.es/p/android-os-processes-and-zygote.html>



Licencia

LIN - Procesos en Android
Versión 0.2

©J.C. Sáez

*This work is licensed under the Creative Commons **Attribution-Share Alike 3.0 Spain License**. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/es/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.*

*Esta obra está bajo una licencia **Reconocimiento-Compartir Bajo La Misma Licencia 3.0 España de Creative Commons**. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.*

Este documento (o uno muy similar) está disponible en
<https://cv4.ucm.es/moodle/course/view.php?id=62472>

