

# Trabajo de Fin de Grado

---

*Creación de un laboratorio virtual para la enseñanza de  
Sistemas Operativos usando Android*

*Autor: Aitor Ardila Bellés*

*Director: Enric Morancho Llena*

# Índice

1. Contexto
2. Estado del arte
3. Alcance
4. Implementación laboratorio virtual
5. Prácticas de laboratorio
6. Demo
7. Conclusiones

# Índice

## 1. Contexto

2. Estado del arte
3. Alcance
4. Implementación laboratorio virtual
5. Prácticas de laboratorio
6. Demo
7. Conclusiones

# 1. Contexto

**Objetivo** → Formar a los estudiantes en los SO de dispositivos móviles

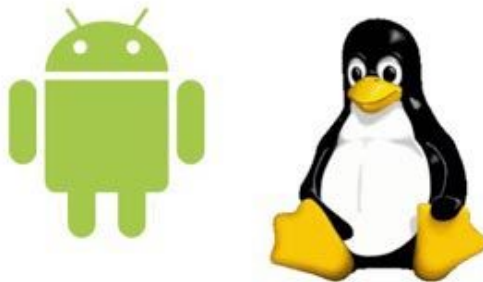
**¿Cómo?** → Creando un lab. virtual para poder trabajar con el código fuente de Android + proponiendo prácticas de laboratorio.



# 1. Contexto

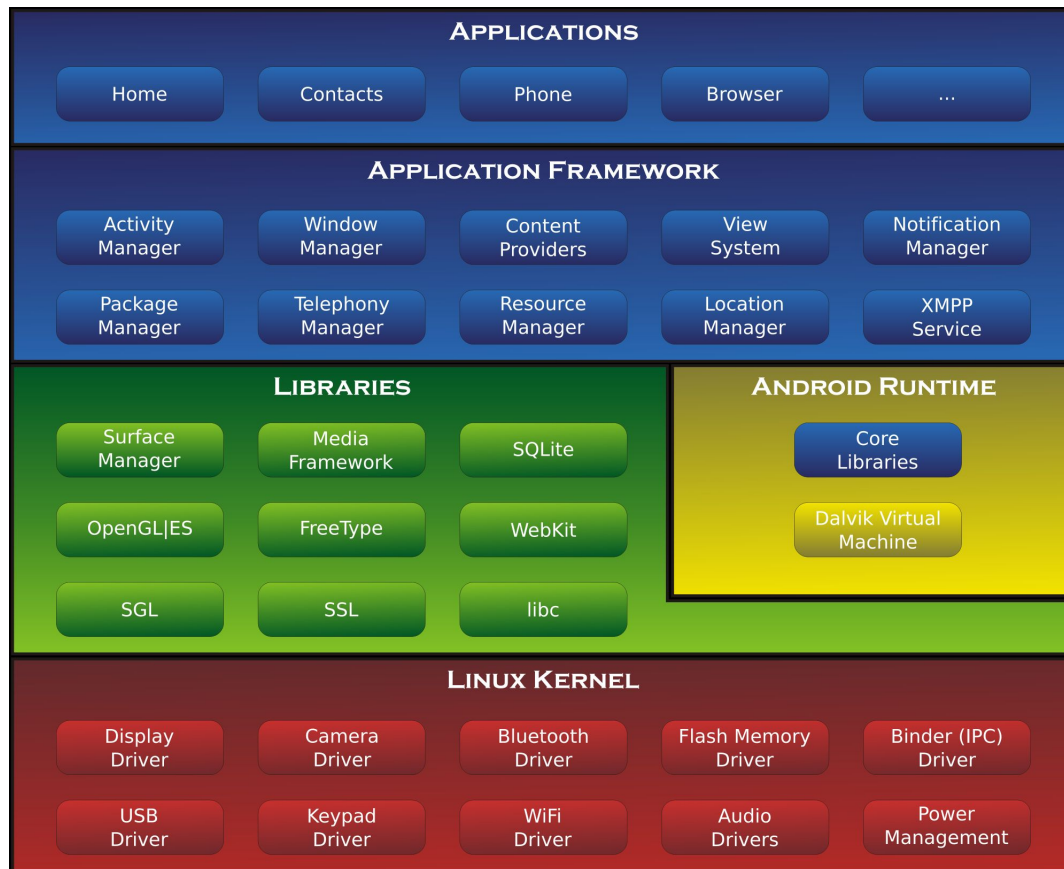
**Android** es un Sistema Operativo basado en el kernel de Linux.

- Adaptado para ser usado en sistemas móviles (ej. *Smartphones*)
- Ofrece:
  - Fácil instalación/desinstalación de aplicaciones
  - Comunicación inalámbrica: *GSMA, CDMA, LTE, Bluetooth, WiFi, NFC*
  - *Wakelocks API*
  - Seguridad adicional
  - ...



# 1. Contexto

Esquema general



# 1. Contexto

Esquema básico de comunicación **Android** - HOST (PC)



- USB
- IP/port

# Índice

1. Contexto
- 2. Estado del arte**
3. Alcance
4. Implementación laboratorio virtual
5. Prácticas de laboratorio
6. Demo
7. Conclusiones



## 2. Estado del arte

En la FIB: **FIB**  UNIVERSITAT POLITÈCNICA  
DE CATALUNYA  
BARCELONATECH

Asignaturas: IDI, CASO, IM ← *estudio de Android*

En el resto de universidades:



← *misma idea conceptual*

**Otros proyectos similares:**



# Índice

1. Contexto
2. Estado del arte
- 3. Alcance**
4. Implementación laboratorio virtual
5. Prácticas de laboratorio
6. Demo
7. Conclusiones

### 3. Alcance

#### Implementación laboratorio virtual

- Análisis de requisitos (HW, SW)
- Estudio de emuladores
- Descarga y compilación Android
- Adición de mejoras
- Empaquetado final

#### Prácticas de laboratorio

1. Instalación del entorno virtual
2. Governor de Android
3. Añadir aplicación de sistema
4. Crear módulo de kernel
5. Sistema de ficheros F2FS

# Índice

1. Contexto
2. Estado del arte
3. Alcance
- 4. Implementación laboratorio virtual**
5. Prácticas de laboratorio
6. Demo
7. Conclusiones

## 4. Implementación lab. virtual

### Análisis de requisitos

- Hardware

Característica	Mínimo	Recomendado	Comentarios
<b>Ordenador</b>			
Arquitectura procesador	64 bits	64 bits	Para versiones de Android inferiores a la 2.3 es posible usar arquitectura de 32 bits.
Espacio disponible disco duro	100 GB	150 GB	En caso de compilar para múltiples dispositivos o usar ccache se recomiendan 150 GB.
Memoria RAM	2 GB	<8GB	Contra más RAM más rápida la compilación. En caso de usar Linux virtualizado, se recomiendan 16 GB de RAM/SWAP.
<b>Teléfono móvil</b>			
Tipo	Android	Android	Todo el proyecto está basado en Android.
<b>Otros</b>			
Cable USB			Dependiendo del dispositivo será tipo microUSB o miniUSB.

## 4. Implementación lab. virtual

- Software (*compilación código fuente*)

Android 6.0 (Marshmallow)		
Característica	Versión	Comentarios
Sistema Operativo	Ubuntu 14.04 LTS Mac OS v10.10 (Yosemite)	No es posible compilar Android en sistemas Windows.
Java Development Kit (JDK)	OpenJDK 8 (Ubuntu) jdk 8u45 (Mac OS)	La versión para Mac OS puede ser la especificada o una superior.
Python	2.6, 2.7	De <a href="http://pythong.org">pythong.org</a>
GNU Make	3.81, 3.82	De <a href="http://gnu.org">gnu.org</a>
Git	1.7 mínima	De <a href="http://git-scm.com">git-scm.com</a>
Librerías y paquetes	bison build-essential curl flex gnupgperf libesd0-dev liblz4-tool libncurses5-devlibsdl1.2-dev libwxgtk2.8-dev libxml2libxml2-utils lzop maven pngcrushschedtool squashfs-tools xsltproc zipzlib1g-dev g++-multilib gcc-multilib lib32ncurses5-dev lib32readline-gplv2-dev lib32z1-dev	Necesarios para la compilación del código fuente en arquitectura x64.

## 4. Implementación lab. virtual

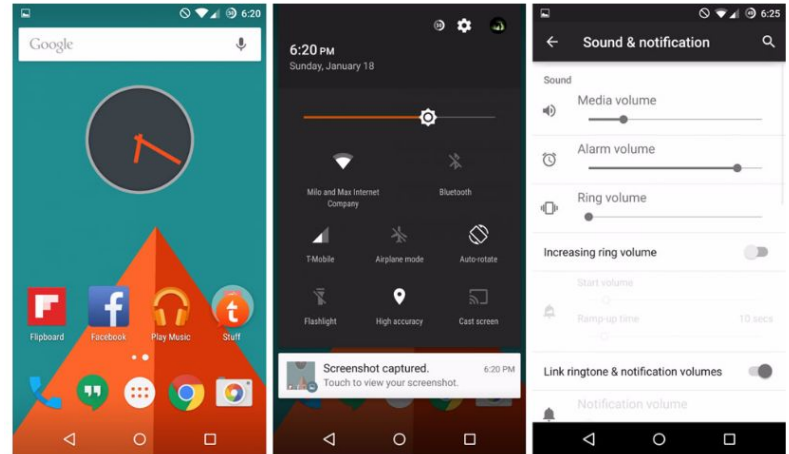
### Preparación del dispositivo físico



1. Desbl. *bootloader*
2. Instalación *recovery*
3. Instalación ROM



cyanogenmod



## 4. Implementación lab. virtual

### Emuladores de Android: Características

Característica / Emulador	AVD	Genymotion	BlueStacks	Android-x86	YouWave
Plataformas	-Windows -Linux -Mac OS X	-Windows -Linux -Mac OS X	-Windows -Mac OS X	-Windows -Linux	Windows
Arquitectura	emulación ARM	x86	x86	x86	x86
Precio	Gratis	Gratis (disponible versión business de pago)	Gratis	Gratis	-Gratis (Android 4.0) -Pago (Android 5.1)
Licencia	Freeware	-Freeware -De pago	Freeware	Open Source (Apache 2.0)	-Freeware -De pago
Permite otro kernel?	Sí	No	No	Sí	Sí
Permite Android 6.0?	Sí	Sí	Sí	Sí	No
Compatible Android SDK?	Sí	Sí	No	Sí	No
Compatible Android Studio?	Sí	Sí	No	No	No
Compatible Eclipse?	Sí	Sí	No	No	No
Modo debug?	Sí	Sí	No	Sí	No
Comunicación dispositivo	-GUI -Comandos	-GUI -Comandos	GUI	-GUI -Comandos	GUI
Cambiar variables estado (batería, GPS, etc.)	Sí	Sí	No	Sí	No
Recomendado para	-Desarrollador -Tester	-Desarrollador -Tester	-No profesional	-Desarrollador -Tester	-No profesional

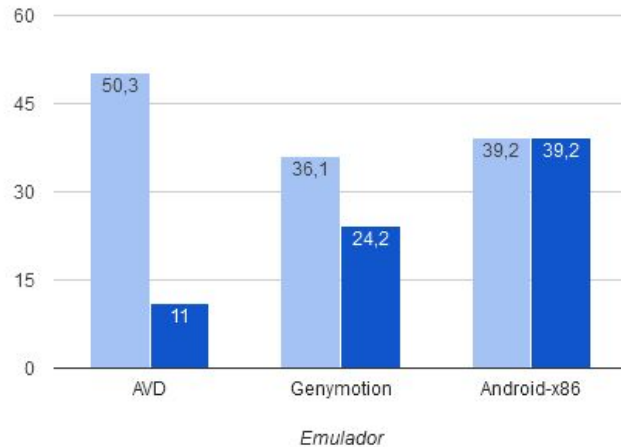


## 4. Implementación lab. virtual

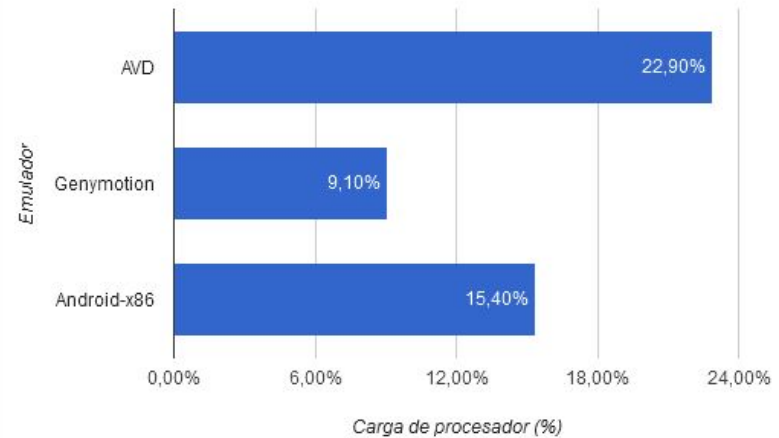
### Emuladores de Android: Comparativa

Tiempo de arranque (s) de Android

1a ejecución    Sigüientes

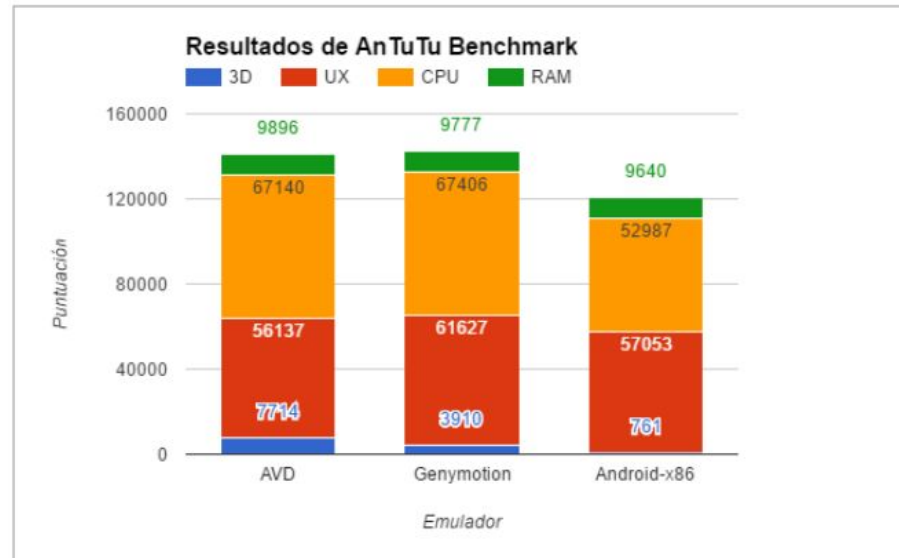
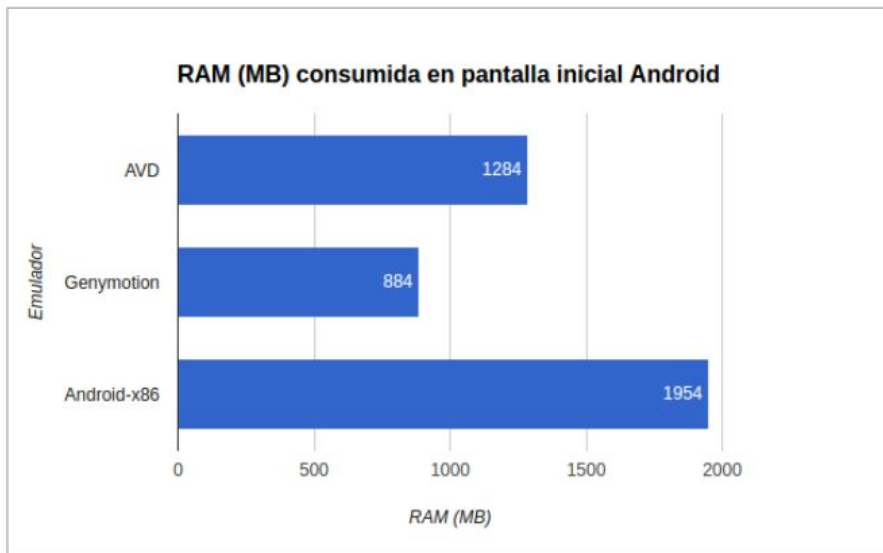


Carga de procesador (%) en pantalla inicial



## 4. Implementación lab. virtual

### Emuladores de Android: Comparativa



## 4. Implementación lab. virtual

### Descarga Android

- Fuente oficial: *Android Open Source Project (AOSP)*
  - Dispositivos oficiales de Google
- Diversas ROMS (ej. *CyanogenMod*)
  - Dispositivos finales predefinidos

android  
open source project



**cyanogenmod**

#### Información

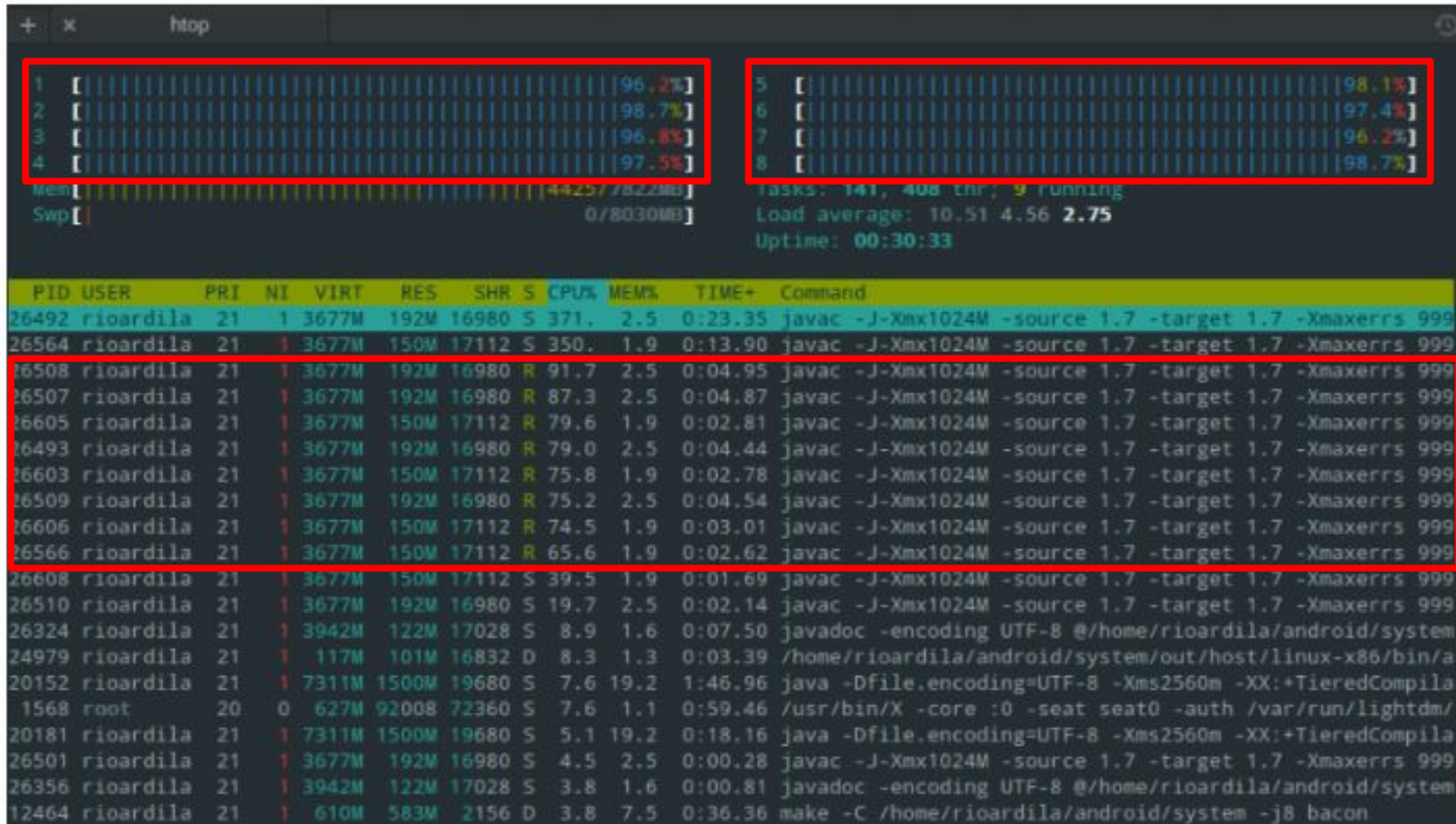
Tamaño: 14,2 GB

Ubicación: /home/rioardila/android/system

## 4. Implementación lab. virtual

### Compilación Android

- Jack toolchain → reduce el tiempo de compilación
  - Compilación incremental
  - Librería .jack con código precompilado
  - Servidor Jack
- Compilación de N threads en paralelo (recomendado:  $N = 2 * \text{threads}$ )
- Opcional: uso de la caché para próximas compilaciones



## 4. Implementación lab. virtual

### Código fuente: Resultado compilación

```
rioardila@marenosturm2:~/android/system/out/target/product/surnia$ ls
android-info.txt          cm_surnia-ota-d46c020717.zip  ramdisk.img
boot.img                  data                          ramdisk-recovery.cpio
cache                     dt.img                       ramdisk-recovery.img
cache.img                 fake_packages                recovery
clean_steps.mk            gen                          recovery.id
cm-13.0-20160928-UNOFFICIAL-surnia.zip  install                     recovery.img
cm-13.0-20160928-UNOFFICIAL-surnia.zip.md5sum  installed-files.txt        root
cm-13.0-20161002-UNOFFICIAL-surnia.zip      kernel                      symbols
cm-13.0-20161002-UNOFFICIAL-surnia.zip.md5sum  obj                         system
cm-13.0-20161104-UNOFFICIAL-surnia.zip      ota_override_device        system.img
cm-13.0-20161104-UNOFFICIAL-surnia.zip.md5sum  ota_script_path            userdata.img
cm_surnia-ota-2e2a883995.zip                 ota_temp
cm_surnia-ota-3fbb1d8f22.zip                 previous_build_config.mk
```

## 4. Implementación lab. virtual

### Ejecución

- En el dispositivo físico:
  - Modo recovery → Factory reset → Flasheado nueva imagen

- En el emulador (AVD):

```
$ emulator -avd MyPhone -system out/target/product/surnia/system.img -ramdisk out/target/product/surnia/ramdisk.img &
```



## 4. Implementación lab. virtual

### Mejoras implementadas

- **Herramientas Cross-Compiling:**  
inclusión de librerías para permitir compilación a arquitectura ARM
- **Script de automatización:**  
Compilación ARM, PUSH al disp., ejecución (+ comprobación resultados parciales)

```
#Choose architecture and compile program
echo "Choose architecture (1. ARM / 2. x86)."
read -r arc
if [$arc -eq 1]; then
    arm-linux-gnueabi-gcc -static "$1" -o "$2"
else if [$arc -eq 2]; then
    gcc "$1" -o "$2"
else
    echo "$(tput setaf 1)Chosen option is not correct"
    tput sgr0
    exit 1
fi

#Check if compilation succeeded
if [ $? -ne 0 ]; then
    echo "$(tput setaf 1)Compilation failed"
    tput sgr0
    exit 1
else
    echo "$(tput setaf 2)Compilation OK"
    tput sgr0
fi
```





## 4. Implementación lab. virtual

### Implementación final del lab. virtual

3 sugerencias:

2. Guía de instalación: pasos necesarios para instalar y configurar todas las herramientas
3. **Guía de instalación con código fuente:** ya descargado y precompilado ← Opción más viable

# Índice

1. Contexto
2. Estado del arte
3. Alcance
4. Implementación laboratorio virtual
- 5. Prácticas de laboratorio**
6. Demo
7. Conclusiones

## 5. Prácticas de laboratorio



### 1. Instalación del laboratorio virtual

Guión con todos los pasos necesarios para su instalación y configuración.

- Basado en la última sugerencia de implementación:
  - Guión con las herramientas necesarias
  - Código fuente ya descargado y precompilado

# 5. Prácticas de laboratorio

## 2. Governor de Android

Cambiar el perfil de comportamiento de la CPU.

```
rioardila@marenosturm2:~/android/system/kernel/motorola/msm8916/drivers/cpufreq$ ls | grep cpufreq_  
cpufreq_conservative.c  
cpufreq_governor.c  
cpufreq_governor.h  
cpufreq_interactive.c  
cpufreq_ondemand.c  
cpufreq_performance.c  
cpufreq_persistent_stats.c  
cpufreq_powersave.c  
cpufreq_stats.c  
cpufreq_userspace.c  
ppc_cbe_cpufreq_pervasive.c  
ppc_cbe_cpufreq_pmi.c
```

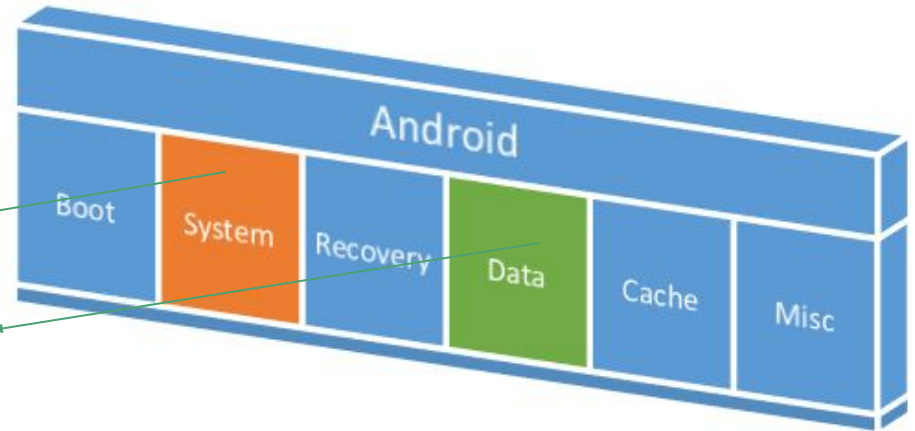


## 5. Prácticas de laboratorio

### 3. Añadir aplicación de sistema

Crear una aplicación de Android y un programa en C y añadirlos en la partición de sistema.

- **System:** SO y apps sistema
- **Data:** info y apps de usuario



## 5. Prácticas de laboratorio

### 3. Añadir aplicación de sistema

> 2 maneras diferentes:

- Fácil: PUSH de la app a SYSTEM
- + interesante: añadirla como parte del código

```
$ adb remount
$ adb push file.apk /system/app/
$ adb shell chmod 644 /system/app/file.apk
$ adb reboot
```

fuente y compilarla

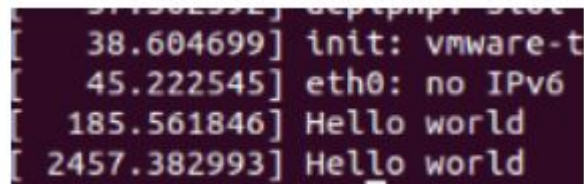
```
rioardila@marenosturm2:~/android/system/packages/apps$ ls
AudioFX          CMFileManager    FMRadio          PackageInstaller  Tag
BasicSmsReceiver CMUpdater        Gallery2         PhoneCommon       Terminal
Bluetooth        CMWallpapers     Gello           Profiles          ThemeChooser
BluetoothExt     Contacts         HTMLViewer      Provision         Trebuchet
Browser          ContactsCommon  InCallUI        Screencast        TvSettings
Calendar         DeskClock       KeyChain         Settings          UnifiedEmail
Camera2          Dialer          LockClock       SetupWizard       WallpaperPicker
CarrierConfig    Eleven          ManagedProvisioning SmartCardService
CellBroadcastReceiver Email           Messaging       Snap
CertInstaller    ExactCalculator  Nfc             SoundRecorder
CMBugReport      Exchange        OneTimeInitializer Stk
```

## 5. Prácticas de laboratorio

### 4. Crear un módulo de Kernel

Creación de un módulo de kernel, compilación, ejecución de la nueva imagen de sistema e instalación del módulo.

```
$ adb push my_module.ko /data/local
$ adb shell
$ cd data/local
$ insmod my_module.ko
$ rmmod my_module
$ dmesg
```



```
[ 38.604699] init: vmware-t
[ 45.222545] eth0: no IPv6
[ 185.561846] Hello world
[ 2457.382993] Hello world
```



## 5. Prácticas de laboratorio

### 5. Sistema de ficheros F2FS

1. Formateo de */data* y */cache* con el sistema de ficheros F2F2, optimizado para memorias flash
2. Tests de benchmark *ext4* vs *F2FS*

Benchmark / Formato	EXT4	F2FS
RL Bench	21,4 s	14,4 s
CF Bench	33478 puntos	34172 puntos
0xBenchmark	1147 puntos	1168 puntos
AnTuTu Benchmark	30134 puntos	31306 puntos
Quadrant Benchmark	9588 puntos	10736 puntos

## 5. Prácticas de laboratorio

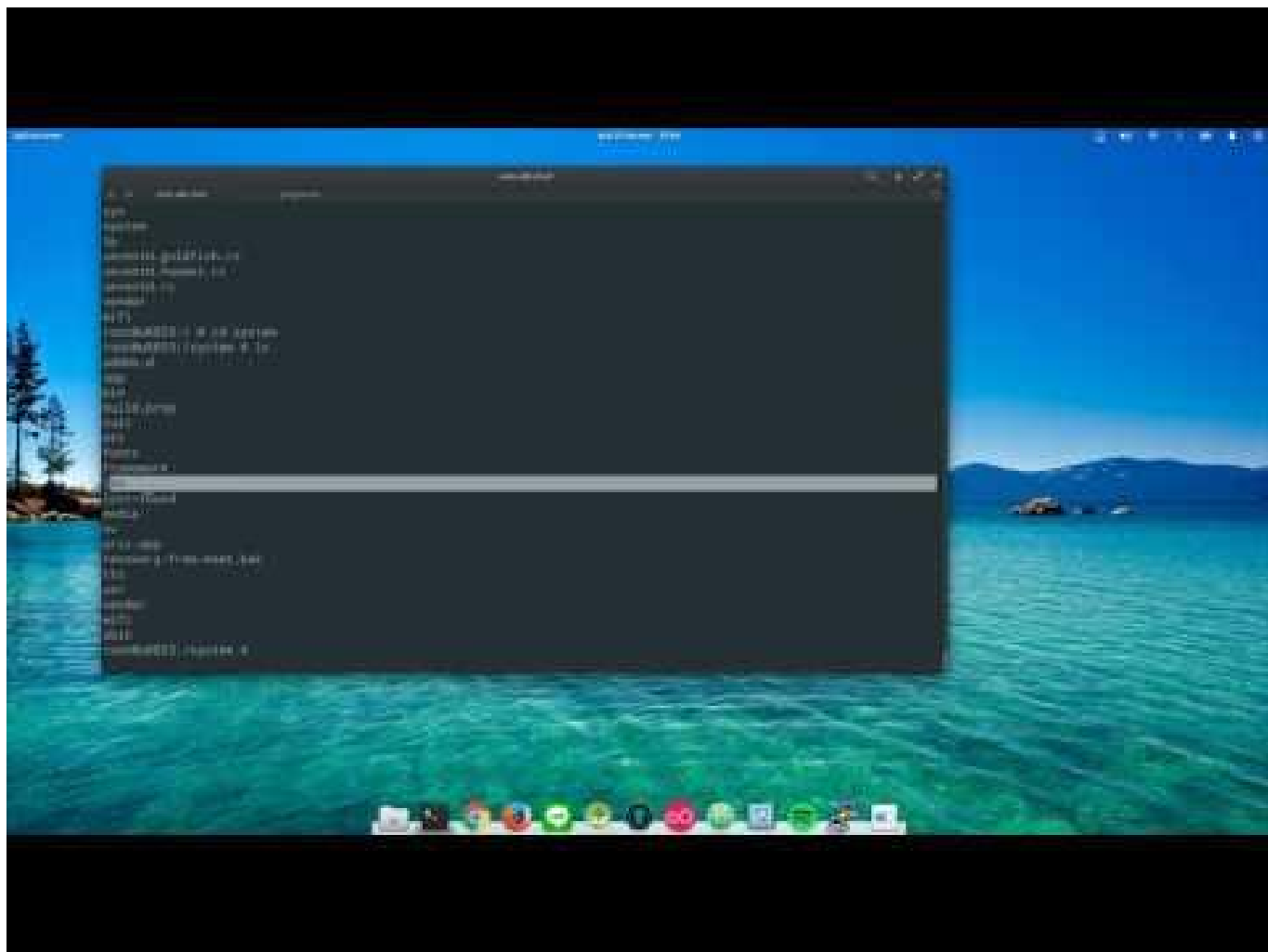
### Análisis valorativo

1. Instalación lab.
  2. Governor
  3. App de sistema
  4. Módulo kernel
  5. Sistema de ficheros
- F2F2

Práctica	Dedicación (h)	Dificultad	Curso recomendado
1	4	Media	2º, 3º
2	2	Alta	4º
3	2	Media	2º, 3º
4	2	Media	2º, 3º
5	2	Baja	2º

# Índice

1. Contexto
2. Estado del arte
3. Alcance
4. Implementación laboratorio virtual
5. Prácticas de laboratorio
- 6. Demo**
7. Conclusiones



# Índice

1. Contexto
2. Estado del arte
3. Alcance
4. Implementación laboratorio virtual
5. Prácticas de laboratorio
6. Planificación y gestión económica
- 7. Conclusiones**

## 7. Conclusiones

- Es interesante usar Android para enseñar Sistemas Operativos
- Es viable de ser usado en un entorno académico
- Simplifica y facilita el trabajo del profesor
- Puede ser usado con cualquier distribución fuente y dispositivo
- Ninguna desviación temporal ni económica

---

*“ Android es uno de lo sistemas más abiertos que he visto jamás. Lo que hace a Android especial es que está literalmente diseñado desde el principio para ser personalizado de una manera muy poderosa. ”*

*Sundar Pichai - Director Ejecutivo de Google Inc.*

## 6. **Planificación** y gestión económica

### Fase inicial

Búsqueda de información + curso de GEP

### Fase de implementación (*metodología Scrum*)

Lab. virtual + prácticas de laboratorio

### Fase final

Redacción Memoria + presentación final



## 6. Planificación y gestión económica

- Costes directos
- Costes indirectos
- Contingencia: 13% del CD y CI
- Imprevistos\*

= **PRESUPUESTO:**

Concepto	Coste (euros)
Costes directos	4630
Costes indirectos	129
Contingencia	618,67
Imprevistos	216,70
<b>Total</b>	<b>5594,37</b>

### \* Imprevistos

- Avería del ordenador
- Avería del móvil
- Retraso de 2 semanas

Imprevisto	Unidades	Probabilidad	Precio (euros)	Coste (euros)
Avería ordenador	1	2%	560	11,20
Avería teléfono móvil	1	5%	0	0
Retraso 2 semanas	75 h	20%	1027,50	205,50
<b>Total</b>			<b>1587,50</b>	<b>216,70</b>