**Práctica 6**  
**PREGUNTAS TEÓRICAS**   
  
**Conceptos general**

1. **¿Qué es Android?**  
   Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets o tabléfonos; y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró.
2. **¿Qué versión del kernel de Linux se está utilizando para desarrollar Android?**  
   //MONOLITICO  
   Android utiliza Linux como kernel o núcleo y en este núcleo está el denominado Dalvik, una máquina virtual de Java creada por Google, es decir, un software que simula una computadora y que permite ejecutar aplicaciones. La diferencia entre el funcionamiento de un smartphone y un PC es que el Dalvik contenido en el kernel de Android ejecuta archivos .dex en lugar de archivos .class, que son los clásicos que ejecuta la máquina virtual de Java en nuestro PC. Los archivos .dex son más compactos y están optimizados para smartphones.

**Android corre sobre Linux (v4.4.1)**

1. **¿Qué aprovecha Android de Linux?(hay que agregar mas)  
   -** Android al provenir de Linux (software libre) permite ser más estudiado que los productos de la competencia (Micrsoft-Windows y Apple-Iphone). Y también manipulado por usuarios avanzados.  
   - **Android se aprovecha de Linux para:**

* **Abstracción de hardware.**
* **Administración de memoria.**
* **Administración de CPU.**
* **Networking.**
* **Seguridad.**

1. **¿Cuál es la mascota de Android? ¿Cuál es su etimología?**La mascota se llama Andy es un robot (Android = androide). está basada en una historia de androides de la novela “¿Sueñan los androides con ovejas eléctricas?” de Philip K. Dick.
2. **¿Cuándo y por quién nació Android?**Android nace en el 2003 Andy Rubin, Rich Miner, Chris White y Nick Sears fundan Android Inc. con el objetivo de desarrollar un sistema operativo para móviles basado en Linux.

1. **¿Cuándo fue comprado por la empresa Google?**En el año 2005 google adquirió Android inc.
2. **¿Cuándo fue presentada su versión beta? ¿Qué es la OHA?**  
   - Al mismo tiempo que se anunciaba la formación de la Open Handset Alliance el 5 de noviembre de 2007, la OHA presentó Android, una plataforma de código libre para teléfonos móviles basada en el núcleo operativo Linux.1 Una beta del SDK fue lanzada para desarrolladores el 12 de noviembre de 2007.3 Basado en una licencia de código libre, compite contra otras plataformas móviles propietarias de Apple, Microsoft, Nokia, Palm, BlackBerry (compañía) y Bada.  
   - La **OHA** es una alianza comercial de 84 compañías que se dedica a desarrollar estándares abiertos para dispositivos móviles.
3. **¿En qué año se lanzó la primera versión estable?**Version 1.0 (Apple Pie) estable 2008.

**Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. Android fue presentado en 2007 junto la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008. Android es el sistema operativo móvil más utilizado del mundo, con una cuota de mercado superior al 80% al año 2017, muy por encima de IOS.**

1. **¿Cuál es la última versión?  
   Android** Pie (**Android** 9.0) - Agosto 2018
2. **¿Con que frecuencia se lanza las versiones del SDK de Android?**
3. Lea <http://www.eazytutz.com/android/android-architecture/> y conteste.  
   **a) ¿Qué rol cumple la capa del kernel de Linux?**  
   es el núcleo de Linux que interactúa con el hardware y que contiene todos los controladores de hardware esenciales. El Kernel de Linux también actúa como una capa de abstracción entre el hardware y otras capas de software.

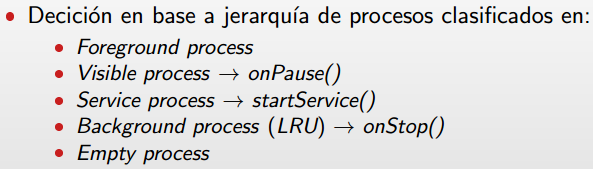
(b) ¿Qué rol cumple la HAL?

**La capa de abstracción de hardware (HAL) brinda interfaces estándares que exponen las capacidades de hardware del dispositivo al framework de la Java API de nivel más alto. La HAL consiste en varios módulos de biblioteca y cada uno de estos implementa una interfaz para un tipo específico de componente de hardware, como el módulo de la cámara o de bluetooth. Cuando el framework de una API realiza una llamada para acceder a hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión.**

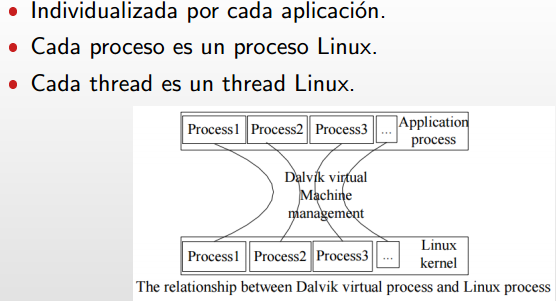
**c) ¿Qué rol cumple la capa de librerías?**  
La siguiente capa es Librerías nativas del Android. Es esta capa que permite que el dispositivo pueda manejar diferentes tipos de datos. Estas librerías son escritas en C o C++ .  
Media framework: diferentes códecs multimedia que permite la grabación y reproducción de diferentes formatos de mediosSQLite:es el motor de base de datos utilizada en el androide con fines de almacenamiento de datosWebKit: Es el motor del navegador se utiliza para mostrar el contenido HTMLOpenGL: sirve para hacer 2D o 3D contenido de gráficos para la pantalla  
  
**d) ¿Qué rol cumple la capa de Android Runtime?**Esta compuesto por Dalvik (máquina virtual) y librerías Core Java.  
  
**Dalvik Virtual Machine:** Es un tipo de máquina virtual utilizada en dispositivos Android optimizada para ejecutar aplicaciones en dispositivos con bajo poder de procesamiento y poca memoria. A diferencia de la JVM no se ejecutan .class, en Dalvik se ejecutan .dex en el momento de la compilación, que proporciona mayor eficiencia en entornos de bajos recursos. La máquina virtual Dalvik permite múltiples instancias de máquina virtual que se creará al mismo tiempo que proporciona la seguridad, el aislamiento, la gestión de memoria y soporte para hilos.  
  
**ART:** Google ha introducido una nueva máquina virtual conocida como ART (Android Runtime) en sus versiones más recientes de Android. En Lollipop, la máquina virtual Dalvik se sustituye completamente por ART. ART tiene muchas ventajas sobre Dalvik VM como AOT (antes de tiempo) la compilación y la mejora de la recolección de elementos que potencian el rendimiento de aplicaciones de manera significativa.  
  
**Core Java Libraries:** Son diferentes librerías de Java SE y Java ME. Sin embargo estas librerías proporcionan la mayor parte de las funcionalidades definidas en las librerías de Java SE  
  
  
**e) ¿Por qué fue reemplazada la DVM? ¿Qué mejora?** ART tiene muchas ventajas sobre Dalvik VM como AOT (antes de tiempo) la compilación y la mejora de la recolección de elementos que potencian el rendimiento de aplicaciones de manera significativa.  
  
**f) ¿Qué rol cumple la capa de Application Framework?**Esta capa es la interactúa directamente con las aplicaciones. Estos programas gestionan las funciones básicas del teléfono como la gestión de recursos, llamadas de voz, etc. Como dearrollador, solamente se considera que son algunas de las herramientas básicas con las que estamos construyendo nuestras aplicaciones.   
Entre los bloques mas importantes están:   
Gestor de actividad (Activity Manager) : Gestiona el ciclo de vida de las aplicaciones.   
Proveedores de Contenido (Activity Manager): gestionar el intercambio de datos entre aplicaciones.  
Director de la telefonía (Activity Manager): Maneja todas las llamadas de voz. Utilizamos gestor de telefonía si queremos acceder a las llamadas de voz en nuestra aplicación.  
Location Manager: Gestión de ubicaciones, a través de GPS o antena de telefonía móvil.  
Administrador de recursos (Resource Manager): Administrar los diferentes tipos de recursos que utilizamos en nuestra solicitud.

**PROCESOS  
1. ¿Qué comandos de GNU/Linux tenemos disponibles en Android?**abcc, adb, am, app\_process, applypatch, atrace, bdaddrwriter, bmgr, bootanimation, bu, bugmailer.sh, bugreport, camera\_fw\_check, cat,chat,chmod.,chown,cluster,cluster\_get.sh,cluster\_set.sh,cmp,content,cp,dalvikvm,date,dbus-daemon,dcc,dd,debuggerd,dexopt,df,dhcpcd,dmesg,dnsmasq,downloader,drmserver  
du,dumpstate,dumpsys,e2fsck,fsck\_msdos,fu,getevent,getprop,glgps,grep,gzip,hd,hdc,p\_test,hostapd,hotplug,id,ifconfig,iftop,ime,input,insmod,installd,ioctl,ionice,ip,ip6tables,iptables,isp\_fw\_update,keystore,kill,linker,ln,log,logcat,logwrapper,ls,lsmod,lsof,make\_,ext4fs,md5,mdnsd,mediaserver,mkdir,mksh,modem\_bridge,monkey,mount,mount\_deb,ugfs.sh,mtpd,mv,nandread,ndc,netcfg,netd,netstat,newfs\_msdos,notify,nv\_hciattach,  
nvtest,ping,pm,pppd,printenv,ps,Python,racoon,reboot,renice,requestsync,rild,rm,rm\_ts,\_server,rmdir,rmmod,route,run-as,scalar\_fw\_update,schedtest,schedtop,screencap,screenshot,sdcard,send\_bug,send,event,sensors-config,sensorservice,service,servicemanager,setconsole,setprop,settings,setup\_fs,sh,sleep,smd,stagefright,start,stop,surfaceflinger,svc,sync,system\_server,tc,tegrastats,tf\_daemon,tinycap,tinymix,tinyplay,toolbox,top,touch,touch\_fw\_update,  
uiautomator,umount,uptime,vdc,vmstat,vold,watchprops,wifimacwriter,wipe,wpa\_suppli,cant,wvdrmserver,xaplay

**2. ¿Android cuenta con un área de intercambio?, ¿Por qué?**Android no cuenta con un área de intercambio (Swap). Es necesario adquirir permisos root y el uso de aplicaciones extras para lograrlo. De todos modos no es tan productivo llevar esto a cabo como en Desktop por cuestiones de hardware.

**3. ¿Qué alternativa tiene a la hora liberar memoria?, ¿Cómo se clasifica los procesos?**Android elimina procesos bajo demanda ante la ausencia de memoria.  


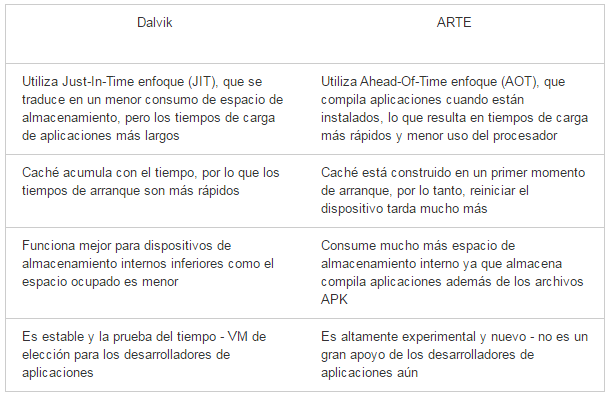
**4. Lea este artículo y responda:**

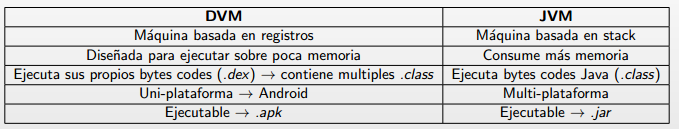
**(a) Por defecto, ¿en cuántos procesos/threads corren los componentes de una aplicación?**En uno.  
  
**Sí, existe la posibilidad y se define mediante android:services. Sí. No, un método o función thread-safe puede correrse sobre multiples hilos asegurandose que los datos usados por uno no son corrompidos por otro, por lo tanto hay que tener cuidado de no modificar el componente de UI desde otro hilo que no sea el principal.**

**(b) ¿Los componentes de las aplicaciones pueden correr en procesos separados y/o en el mismo proceso?**Cuando una aplicación se inicia, lo que sucede de manera muy técnica, es que se crea un proceso. Entonces todo lo que se ejecuta dentro de esa aplicación es parte del mismo proceso de ejecución. Un hilo es otra forma de dividir el trabajo, un proceso puede tener varios hilos de ejecución, el hilo lo podemos tomar como una secuencia de control dentro de cualquier proceso y este ejecuta lo que indiquemos de manera independiente.

**(c) ¿Se pueden crear n threads de un proceso?** Sí .De hecho es muy común que esto suceda. Con el fin de no bloquear el hilo principal.

**(d) ¿Los componente de la vista de una aplicación son thread-safe ?, ¿Qué reglas sigue el Android’s single thread model ?**Cuando un usuario ejecuta una aplicación, Android crea un hilo de ejecución que se denomina principal (main). Este hilo es muy importante porque es el encargado de gestionar los eventos que dispara el usuario a los componentes adecuados e incluye también los eventos que dibujan la pantalla. Por esta razón, a este hilo principal también se le llama hilo de la interfaz de usuario (UI thread).  
Este modelo de ejecución de las aplicaciones se denomina Modelo de ejecución de un único hilo (Single Thread Model).  
La interfaz de usuario de Android no es thread-safe. Una función o método trhead-safe puede ser invocado por múltiples hilos de ejecución sin preocuparnos de que los datos que accede dicha función (o método) sean corrompidos por alguno de los hilos ya que se asegura la atomicidad de la operación, es decir, la función se ejecuta de forma serializada sin interrupciones.  
Es decir, en Android se puede manipular la interfaz de usuario desde otro subproceso. Por lo tanto, hay que tener en cuenta dos reglas a la hora de diseñar aplicaciones en Android:  
 1. No bloquear nunca el hilo principal o de la interfaz de usuario.  
 2. No acceder a la interfaz de usuario de Android desde un hilo exterior.

**5. ¿Qué es DVM ?, ¿Por qué fue diseñada?. Compare este concepto con el de JVM. Adicionalmente compare el concepto de DVM con el de ART en base a …….. artículo.  
  
Dalvik Virtual Machine:** Es un tipo de máquina virtual utilizada en dispositivos Android optimizada para ejecutar aplicaciones en dispositivos con bajo poder de procesamiento y poca memoria. A diferencia de la JVM no se ejecutan .class, en Dalvik se ejecutan .dex en el momento de la compilación, que proporciona mayor eficiencia en entornos de bajos recursos. La máquina virtual Dalvik permite múltiples instancias de máquina virtual que se creará al mismo tiempo que proporciona la seguridad, el aislamiento, la gestión de memoria y soporte para hilos.  
En principio fue diseñada por problemas de licencia por el uso de la JVM. En el diseño de la reemplazante (DVM) se buscaron/encontraron muchas mejoras. Luego esta fue reemplazada recientemente por ART



**6. ¿Qué es y de qué se encarga Zygote?**Android está divido en varias capas o niveles: capa Android (niveles Application, Framework, Runtime y Libraries) y Linux. **Zygote** escucha en un determinado socket a la espera de recibir comandos y genera un nuevo proceso bifurcándose ('fork') como un proceso de la capa Linux, fuera de las capas Android.

Los diseñadores de Android prefirieron, muy acertadamente, ejecutar cada aplicación como un proceso diferente y completamente aislado. Dejaron esta parte de la gestión de procesos al núcleo Linux, que es muy eficiente y está más que probado. Con esta arquitectura de máquinas virtuales replicadas en procesos independientes, se generan otras infeficiencias: cada proceso debe cargar su máquina virtual, y éste debe cargar después todas las clases estándar de Java que hacen falta para la aplicación. Esto haría que el tiempo de arranque de cada programa fuese inaceptable, y han recurrido a un truco muy guapo para solucionarlo (por eso digo que la arquitectura de Android es una obra impresionante de ingeniería informática): el zygote.

Android crea un proceso inicial, el zygote, que carga la máquina virtual y las librerías estándares de Java y del API de Android. Cada nueva aplicación que se arranca es hija de este zygote, y ya ganaron eficiencia y se ahorra muchísima memoria.  
  
**APLICACIONES**

1. ¿Qué componentes de una aplicación Android existen? ¿para que sirve cada uno?

**Componentes: activity, service, rceiver, provider.**

**Cuando se invoca el primer componente de una aplicacion se crea un proceso Linux con un unico thread (main/UI thread).**

**Por defecto todos los componentes de una aplicacion se ejecutan sobre ese mismo thread del proceso que represente a la aplicacion.**

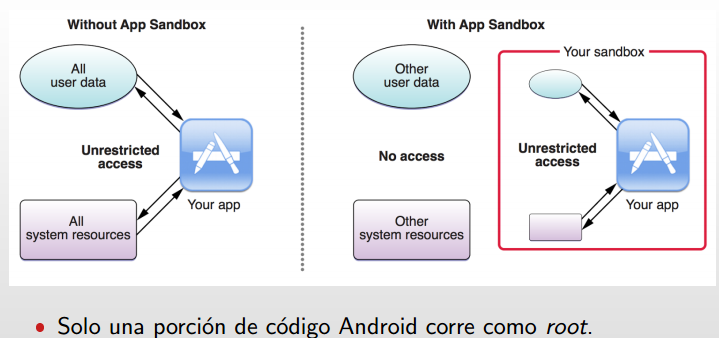
**Los componentes pueden ser configurados para que ejecuten sobre diferentes procesos (android:process).**

**Es posible crear threads dentro de cada proceso → importante para el acceso a la UI (UI toolkit thread unsafe).**

**2. ¿Qué es Gradle?, ¿Qué tarea permite realizar?**Gradle es una herramienta para automatizar la construcción de nuestros proyectos, por ejemplo las tareas de compilación, testing, empaquetado y el despliegue de los mismos. Es muy flexible para la configuración, pero además ya tiene armadas las tareas para la mayoría de los proyectos por default. Esta herramienta es usada por grandes proyecto “Open Source” como “Spring”, “Hibernate”, y “Grails”.

3. ¿A qué hacen referencia y para qué sirven las varibles compileSdkVersion, minSdkVersion,  
targetSdkVersion y buildToolsVersion? ¿tienen que tener alguna relación entre ellas?. ¿Y la  
variable versionCode? ¿que impacto tiene en la distribución de una aplicación a través de  
Google Play Store?

**4. ¿Todas las aplicaciones tiene que estar firmadas digitalmente para ejecutar en un dispositivo? ¿A través de que mecanismo se realiza la firma digital? ¿A través de que herramienta se lo puede hacer?**Todas las aplicaciones (.apk) deben ser firmadas digitalmente.  
Se emiten certificados auto-firmados (KeyTool):  
 Debug mode-> debug key -> el keystore se crea automáticamente ($HOME/.android)  
Release mode -> developer’s private key -> manualmente.

**5. ¿Qué es el application sandbox?**Es un mecanismo para ejecutar programas con seguridad y de manera separada

**El application sandbox es lo que permite la seguridad en las aplicaciones android.**

**A nivel de AndroidManifest → permisos a recursos que termina de conceder o configurar el usuario (aplicado por el sandbox).**

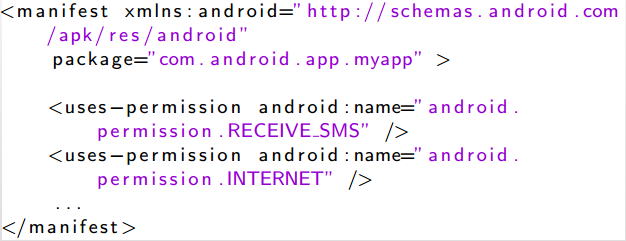
**A nivel de las aplicaciones y sus archivos → certificado, userId y permisos UGO (aplicado por el sandbox).**

Es un enfoque para el desarrollo de aplicaciones que limita el ambiente donde la app se va a ejecutar. Es una forma de seguridad para que la aplicación no interactue (o interactue como queramos) con cierta informacion del telefono.

Solo una porcion del codigo Android corre como root, las apps corren en modo usuario.

**6. ¿Existe la posibilidad de que dos aplicaciones compartan el userId? ¿Qué debe respetarse?**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.  
Las aplicaciones que comparten el userId tienen que compartir la firma. Es decir que deben ser firmados por la misma clave privada.

**7. ¿Qué alternativas tienen las aplicaciones para compartir sus datos?**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.

**8. ¿Dónde se define el acceso a los recursos por parte de las aplicaciones?, ¿Pueden realizarse cambios en tiempo de ejecución?, ¿Qué excepción se lanza si una aplicación intenta acceder a un recurso sobre el cual no definió permisos?**Se debe declarar el acceso a los recursos de manera estática (mainfest.xml) --> dinámico a partir de v6.0 Cuando la aplicación es instalada el usuario debe dar su consentimiento.  
  
  
La excepción que se lanza es SECURITY EXCEPTION .

**9. ¿En qué momento el usuario le da permiso a la aplicación para que esta utilice los recursos del dispositivo? ¿Se pueden definir de manera dinámica?**Cuando la aplicación es instalada. Si a partir de la v6.0.

**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.**

**Si una aplicación intenta acceder a un recurso sobre el cual no definió permisos se lanza una SecurityException**

La declaración del acceso a los recursos paso a ser dinámico a partir de v6.0. Cuando la aplicación es instalada el usuario debe dar su consentimiento.

10. ¿Qué es un y qué contiene un APK?

11. ¿Para qué sirve el archivo AndroidManifest.xml?. ¿Qué son los archivos .dex?

**Un archivo con extensión .apk (Android Application Package) es un paquete para el sistema operativo Android. Este formato es una variante del formato JAR de Java y se usa para distribuir e instalar​ componentes empaquetados para la plataforma Android**

**Un archivo APK es un archivo que normalmente contiene los siguientes archivos y directorios:**

**META-INF directorio:**

**MANIFEST.MF: el Archivo manifest**

**CERT.RSA: El certificado de la aplicación**

**CERT.SF: la lista de recursos y Digestor de las líneas correspondientes en el archivo MANIFEST.MF**

**armeabi-v7a: Código de compilación sólo para todos los procesadores ARMv7 y superiores**

**arm64-v8a: Código compilado para todos los procesadores ARMv8 arm64 y superiores4​5​**

**x86: Código compilado sólo para procesadores x86**

**x86\_64: Código compilado sólo para procesadores x86 64**

**mips: Código compilado sólo para procesadores MIPS**

**res: El directorio que contiene recursos no compilados en resources.arsc (ver más abajo).**

**assets: Un directorio que contiene recursos de aplicaciones, que puede ser recuperado por**

**AssetManager.**

**AndroidManifest.xml: Un archivo de manifiesto adicional de Android, que describe el nombre, la versión, los derechos de acceso y los archivos de bibliotecas referenciados para la aplicación. Este archivo puede estar en Android binario XML Que se pueden convertir en XML de texto claro legible por humanos con herramientas tales como AXMLPrinter2, android-apktool, o Androguard.**

**classes.dex: Las clases compiladas en el formato de archivo dex Comprensible por el Dalvik virtual machine**

**resources.arsc: Un archivo que contiene recursos precompilados, como XML binario, por ejemplo.**

12. Detalle el proceso de firma (signing) de un APK.

**Puede ser de manera manual, o bien a traves de Android Studio**

[**https://developer.android.com/studio/publish/app-signing**](https://developer.android.com/studio/publish/app-signing)

**Almacenamiento**

**1. ¿Qué alternativas tiene una aplicación para almacenar sus datos? Detalle cada una.**  
Existen muchas alternativas para almacenar información de forma permanente en un sistema informático. A continuación mostramos una lista de las más habituales utilizadas en Android:

Preferencias: Es un mecanismo liviano que permite almacenar y recuperar datos primitivos en la forma de pares clave/valor. Este mecanismo se suele utilizar para almacenar los parámetros de configuración de una aplicación.

Ficheros: Puedes almacenar los ficheros en la memoria interna del dispositivo o en un medio de almacenamiento removible como una tarjeta SD. También puedes utilizar fichero añadidos a tu aplicación como recursos.

XML: Se trata de un estándar fundamental para la representación de datos, en Internet y en muchos otros entornos (como en el Android SDK). En Android disponemos de las librerías SAX y DOM para manipular datos en XML.

Base de datos: Las APIs de Android contienen soporte para SQLite. Tu aplicación puede crear y usar base de datos SQLite de forma muy sencilla y con toda la potencia que nos da el lenguaje SQL.

Proveedores de contenidos: Un proveedor de contenidos es un componente opcional de una aplicación que expone el acceso de lectura / escritura de sus datos a otras aplicaciones. Está sujeto a las restricciones de seguridad que quieras imponer. Los proveedores de contenido implementan una sintaxis estándar para acceder a sus datos mediante URI (Uniform Resource Identifiers) y un mecanismo de acceso para devolver el dato similar a SQL. Android provee algunos proveedores de contenido para tipos de datos estándar, tales como contactos personales, ficheros multimedia, etc.

Internet: Además también puedes usar la nube para almacenar y recuperar datos.

**2. ¿En dónde se almacenan las shared preferences de las aplicaciones? ¿De qué tipo pueden ser? ¿Una aplicación podría acceder a las shared preferences de otra diferente?**Se guardan en un archivo XML en forma de clave-valor, y se guardan en el sistema de archivos de la siguiente forma:

/data/data/com.namespace.de.nuestra.aplicacion/shared\_prefs/nombre\_de\_nuestras\_prefs.xml

Se almacenan en /data/data/<package name>/shared\_prefs/. Son archivos XML que generalmente suelen almacenar por ejemplo la configuracion de las aplicaciones, las preferencias que los usuarios setean (se almacena clave-valor).

Pueden ser privadas (MODE PRIVATE) o compartidas (MODE WORLD READABLE o MODE WORLD WRITEABLE).

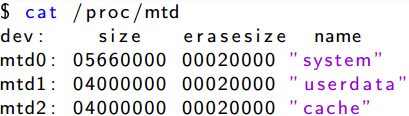
**3. ¿En dónde se almacenan las bases de datos de las aplicaciones?**Todas las aplicaciones de la root (o no) tener un directorio de datos predeterminado, que es /data/data/<package\_name>. De forma predeterminada, las aplicaciones de bases de datos, configuración, y todos los otros datos, vaya aquí. Si una aplicación de espera enormes cantidades de datos a ser almacenados, o por otras razones, quiere "ser agradable para almacenamiento interno", hay un directorio correspondiente en la SDCard (Android/data/<package\_name>).

Se almacenan en /data/data/<package name>/databases/

**File system**

**1. ¿Cuáles son los puntos de montaje principales del File System de Android?, ¿Que contiene cada uno?**La jerarquía de archivos Android es una versión modificada de la jerarquía de ficheros Linux tradicional. Existen ligeras variaciones en la estructura a través de la versión diferente de Linux, y de diferentes fabricantes. Sin embargo, las variaciones son leves. A continuación se presenta un breve resumen de los niveles superiores de la estructura de directorios para la versión AOSP de Jelly Bean:  
  
**/system** - Este directorio es el punto de montaje para el /dev/block/mtdblock0. Bajo este directorio son los directorios que normalmente vemos en el directorio raíz de una distribución Linux estándar. Estos directorios incluyen bin, etc, lib, usr, y xbin.  
Esta partición contiene, básicamente, todo el sistema operativo, que no sea el núcleo y el disco RAM. Esto incluye la interfaz de usuario Android, así como todas las aplicaciones del sistema que vienen pre-instalados en el dispositivo.  
 **/recovery** - La partición de recuperación puede ser considerado como una partición de arranque alternativa que le permite arrancar el dispositivo en una consola de recuperación para la realización de operaciones de recuperación y mantenimiento avanzado en él.  
 **/data -** También llamada de datos de usuario, la partición de datos contiene los datos del usuario - aquí es donde los contactos, mensajes, configuraciones y aplicaciones que haya instalado marcha. Limpiando esta partición esencialmente realiza un restablecimiento de fábrica en el dispositivo, su restauración en la forma en que fue la primera vez que ha arrancado, o la forma en que fue después de la última instalación oficial o ROM personalizada.  
  
**/boot** - Esta es la partición que permite que el teléfono para arrancar. Incluye el núcleo y el disco RAM. Sin esta partición, el dispositivo simplemente no es capaz de arrancar.  
  
**/cache** - Esta es la partición en tiendas de Android utilizan frecuentemente los componentes de datos y aplicaciones. Si se limpia la caché no afecta a sus datos personales, sino que simplemente se deshace de los datos existentes allí, que se reconstruirá automáticamente a medida que continúe el uso del dispositivo.

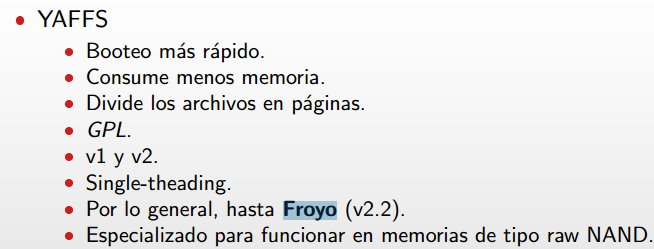
* **/system**: Donde tenemos información correspondiente al sistema
* **/system/bin:** Aplicativos binarios (los comandos)
* **/data/app:** También llamada de datos de usuario, la partición de datos contiene los datos del usuario - aquí es donde los contactos, mensajes, configuraciones y aplicaciones que haya instalado marcha. Los .apk, solo los archivos, no los archivos de la aplicación son almacenados tambien aca.
* **/data/data/<package identifier>:** Aca si estan los archivos de la aplicación en si.
* **/recovery:** Particion que provee la empresa que construye los dispositivos para realizar un recovery ante un problema.
* **/boot:**
* **/cache:** Archivos temporales

**2. ¿Con qué tipos de memoria cuenta un dispositivo móvil generalmente? Detalle cada uno y luego relaciónelos con los tipos de file system que cada tipo podría soportar.**Raw NAND flash:  
 Subsistema MTB.  
 /dev/block/mtdblockN  
   
SD, MicroSD y eMMC:  
 Drive mmcblock.  
 /dev/block/

Hay dos tipos de memorias que se pueden utilizar:

**Raw NAND flash**:Son manejadas a traves de un subsistema que provee linux llamado MTD. Se pueden encontrar en /dev/block/mtdblockN (el numero es la particion).

**SD, MicroSD y eMMC(Son memorias que vienen integradas en el dispositivo)**: Lo manejan Driver mmcblock. Las podemos encontrar en /dev/block/mmcblk[chip number]p[partition number] (el primer numero es el numero de chip, podemos tener una integrada y una externa por ejemplo, y el otro es el numero de particion dentro de esa memoria.)

**3.** Detalle las caracteristicas del YAFFS. **¿Analice las principales características del file system utilizado hasta, generalmente, la versión 2.2 de Android? Nota: licencia, versiones y comparación entre ellas, tratamiento de archivos, theading, etc.**

El filesystem utilizado hasta esa version era **YAFFS.** Booteo más rápido porque fue optimizado para este tipo de memorias, consume menos memoria, divide los archivos en páginas. Corre bajo licencia GPL. Tiene Single-threading por lo cual se reemplaza despues. Por lo general, subsitio hasta Froyo (v2.2). Especializado para funcionar en memorias de tipo raw NAND.

**4. ¿Qué tipo de file system se tiende a utilizar desde la versión 2.3 de Android?, ¿Por qué?**El file system que utiliza Android a partir de la versión 2.3 es EXT4. La mayoría de los dispositivos Android anteriormente usaban YAFFS, que es un sistema de archivos diseñado específicamente para almacenamiento flash. Su problema es que es de un solo subproceso - es decir, que no verá un aumento de rendimiento en esos droides de doble núcleo se rumorea que tan a menudo adornan nuestra sección de noticias últimamente.  
Ext4 ofrece un aumento de rendimiento incluso en las CPU de un solo núcleo, pero con dos núcleos de la diferencia será aún más pronunciado.

**Licencia**

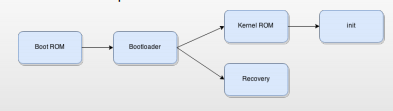
**1. ¿Bajo qué licencias está el stack de Android?, ¿Por qué?**Google decidió mantener GPL fuera del espacio de usuario:   
 Apache v2para el espacio de usuario.  
 GPL para el espacio de kernel.  
 Además de tamaño y optimización, esta fue otra de las razones por las que implementaron su propia versión de libc (BIONIC ). **2. ¿Por qué compañías como Samsung pueden cambiar la interface de sus propias versiones de Android?**Se trata de una capa de personalización implantada en los sistemas operativos Android de Google cuyo objetivo es dotar a los dispositivos de una apariencia modificada a la ya suministrada por el sistema operativo en su versión de fábrica. TouchWiz es utilizado internamente por Samsung para teléfonos con funciones sofisticadas (smartphones) y tabletas, y no está disponible en licencia para terceras partes.

**3. ¿Cómo se llama la libc de Android?, ¿Por qué fue implementada?**La nueva librería que se creó para Android BIONIC es más optimizada y de menor tamaño.

Además de tamaño y optimización la libc fue reeimplementada por una cuestion de licenciamiento, porque libc corre bajo licenciamento GPL y como la libc es usada en el espacio de usuario, la reimplementaron para poder licenciarlo bajo apache2. Esta fue otra de las razones por las que implementaron su propia versión de libc (BIONIC).

**Rooteo**

**1. ¿Qué significa rootear un dispositivo Android?**Darle permisos de root al dispositivo. Es diferente a flashear (pisar el /boot con otra ROM).

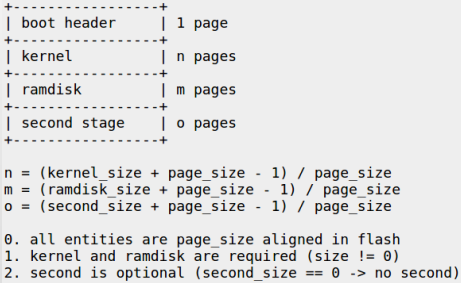
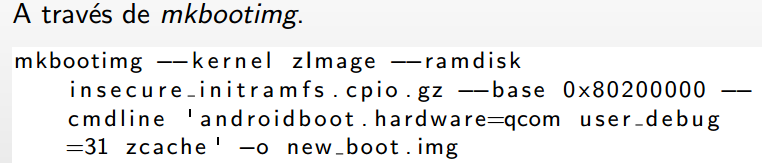
**2. ¿Qué es el bootloader?, ¿De qué se encarga?, ¿Qué tipos existen?, ¿Qué consecuencias trae desbloquearlo?**El bootloader es el encargado de inicializar el hardware y carga el initramfs. Permite bootear una imagen ya sea Android o recovery. Esta fuera de lo que es Android.  
  
Existen distintos tipos de bootloader:  
 Open bootloader: las compañías no lo desean.  
 Locked bootloader: Carrier ID.  
 Conditionally open bootloader: pérdida de garantía.  
Al desbloquearlo se pierde la garantía del proveedor/fabricante impuesto por Digital Rights Management.

El bootloader inicializa el hardware y carga el initramfs. Nos permite bootear una imagen Android o un recovery. Esta fuera de lo que es Android porque cada fabricante lo implementa o utiliza el bootloader que quiere. Desbloquearlo puede implı́car perder la garantı́a del dispositivo (impuesto por Digital Rights Management).

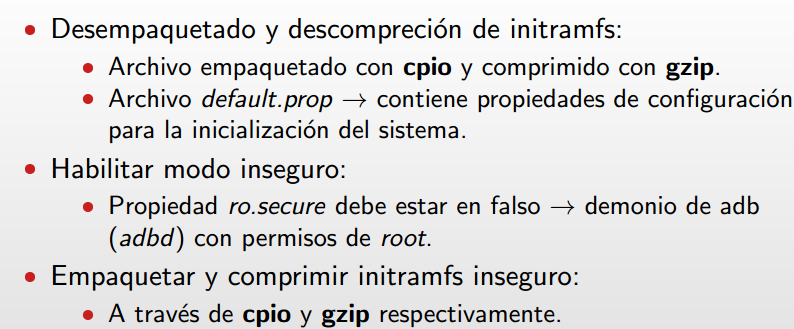
**3. Un bootloader bloqueado permite cargar un sistema operativo particular, ¿Cómo se realiza esta identificación?**Hay distintos tipos de bootloader:

* Open bootloader → las compañias no lo desean.
* Locked bootloader → No te da la opcion de desbloquearlo. Generalmente utilizan un Carrier ID.
* Conditionally open bootloader → pérdida de garantı́a.

**4. ¿Qué es el fastboot ?, ¿Qué acciones permite realizar?**Fastboot es un protocolo usb que permite bootear customs ROMs. Permite flashear una partición. Y además desbloquear el bootloader (solo si el dispositivo lo soporta).

**5. ¿Qué es la boot.img ?, ¿Con qué herramienta se la puede dividir y con cuál crear?, ¿Qué contiene?, ¿Qué nombre tiene la imagen del kernel de Linux ?, ¿En qué formato está empaquetado y comprimido el initramfs?**boot.img contiene el núcleo y ramdisk, los archivos críticos necesarios para cargar el dispositivo antes de que el sistema de ficheros se pueda montar.  
La herramienta que podemos utilizar para dividir es unmkbootimg.  
  
Y para crear….  
  
El kernel de Linux   
Con la ***serie 2.6*** del núcleo, el sistema de numeración así como el modelo de desarrollo ha cambiado. Las versiones han pasado a numerarse con 4 dígitos y no existen versiones de producción y desarrollo.

* Las versiones del núcleo se enumeran hoy en día con 4 dígitos, de la siguiente forma: AA.BB.CC.DD.

AA: Indica la serie/versión principal del núcleo.  
BB: Indica la revisión principal del núcleo. Números pares e impares no tienen ningún significado hoy en día.  
CC: Indica nuevas revisiones menores del núcleo. Cambia cuando nuevas características y drivers son soportados.  
DD: Este digito cambia cuando se corrigen fallos de programación o fallos de seguridad dentro de una revisión.  


**Boot.img** para obtenerla podemos realizar la extracción de una imagen original (tener permisos de root es necesario). Descarga desde un sitio confiable (CyanogenMod).

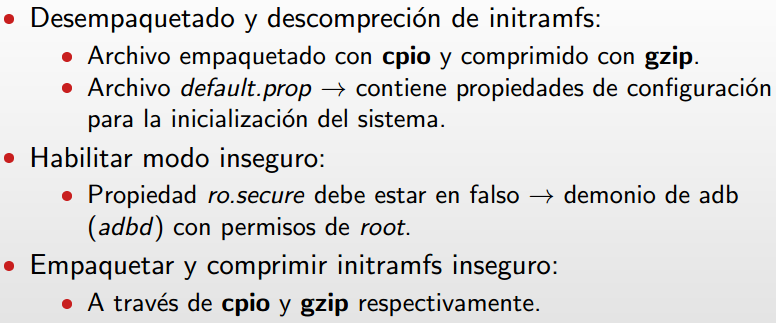
Luego de eso es neceario realizar una division para despues poder acceder al initramfs, a traves de la herramienta unmkbootimg que termina desglosando esa imagen.

El initramfs esta empaquetado con cpio y comprimido con gzip.

**6. ¿Para qué sirve el archivo default.prop que está dentro del** initramfs **?, ¿Qué significa el prefijo ro? ¿A qué hace referencia la propiedad ro.secure?**

El archivo default.prop contiene propiedades de configuración para la inicialización del sistema. Una de esas propiedades es la que hay que “tocar” para habilitar modo inseguro para tener privilegios de root, esa propuedad es ro.secure que debe estar en falso (tiene que ver con el demonio de adb).(adbd) con permisos de root.

El prefijo ro es que es un archivo de solo lectura, no se puede modificar en tiempo de ejecución.

  
//

http://mellynce.soup.io/post/143631165/Android-Property-System

\* Special Properties\* If a property’s name begins with “ro.”, then this property is treated as a read-only property. Once set, the value of the property can’t be changed.

<https://seasonofcode.com/posts/how-rooting-works-a-technical-explanation-of-the-android-rooting-process.html>

ADB (see [the official documentation for ADB](http://developer.android.com/guide/developing/tools/adb.html)) allows a PC or a Mac to connect to an Android device and perform certain operations. One such operation is to launch a simple shell on the device, using the command adb shell. The real question is what user do the commands executed by that shell process run as. It turns out that it depends on the value of an Android system property, named ro.secure. (You can view the value of this property by typing getprop ro.secure either through an ADB shell or on a terminal emulator on the device.) If ro.secure=0, an ADB shell will run commands as the root user on the device. But if ro.secure=1, an ADB shell will run commands as an unprivileged user on the device. Guess what ro.secure is set to on almost every stock OEM Android build. But can we change the value of ro.secure on a system? The answer is no, as implied by the ro in the name of the property. The value of this property is set at boot time from the default.prop file in the root directory. The contents of the root directory are essentially *copied* from a partition in the internal storage on boot, but you cannot write to the partition if you are not already root. In other words, this property denies root access via ADB, and the only way you could change it is by gaining root access in the first place. Thus, it is secure.