Kernel y Native Userspace

LIN - Curso 2015-2016

J.C. Sáez
High Performance Computing Division ArTeCs Group
Universidad Complutense de Madrid

Contenido

Contenido

Introducción

Antes de empezar

1. Iniciar sesión con **Usuario VMs**
2. Descargar 3 ficheros en `~/Descargas`
   - Este documento (campus)
   - FicherosAndroid.tar.gz (campus)
   - Android NDK (En Google Drive - enlace en el campus)

Dirección IP MV Android-x86

192.168.206.134

Introducción

Objetivos

- Aprender a compilar y desplegar módulos del kernel en Android
- Familiarizarse con el proceso de construcción de programas nativos usando GCC y Android NDK
- Familiarizarse con el shell de Android y con los comandos básicos de ADB

Entorno de trabajo

- Usaremos las 2 máquinas virtuales de la asignatura arrancadas simultáneamente con usuario_vms:
  1. Debian: *host* de desarrollo
     - Aumentaremos recursos de la MV por comodidad
  2. Android-x86: *target*

Preparación del entorno de trabajo

Ficheros Necesarios

- Android NDK (link)
  - Almacenar en ~/Descargas
- FicherosAndroid.tar.gz (Campus Virtual)
  - Almacenar en ~/Descargas
- Fuentes de Android-x86 (link)
  - `/mnt/discoVMs/LINyAISO/kitkat-src-norepo.tar.gz`

Otros recursos

- Tar.gz original de la máquina virtual de Debian (link)
  - `/mnt/DiscoVMs/LINyAISO/*c201516-Debian_7_*.tar.gz`
- Tar.gz original de la máquina virtual de Android-X86 (link)
  - `/mnt/DiscoVMs/LINyAISO/Android_x86_LIN.tar.gz`

Preparación del entorno de trabajo

Pasos

1. Lanzar la MV de Debian
2. Modificar configuración de MV de Debian y reiniciar la MV
   - Aumentar # de cores
   - Añadir carpetas compartidas
   - ... y reiniciar
3. Descomprimir código de Android-x86 y FicherosAndroid.tar.gz en MV
4. Instalar Android NDK en MV
5. Liberar espacio en el disco de la MV

Modificar configuración de la MV de Debian

- Arrancar MV de Debian (desde terminal)

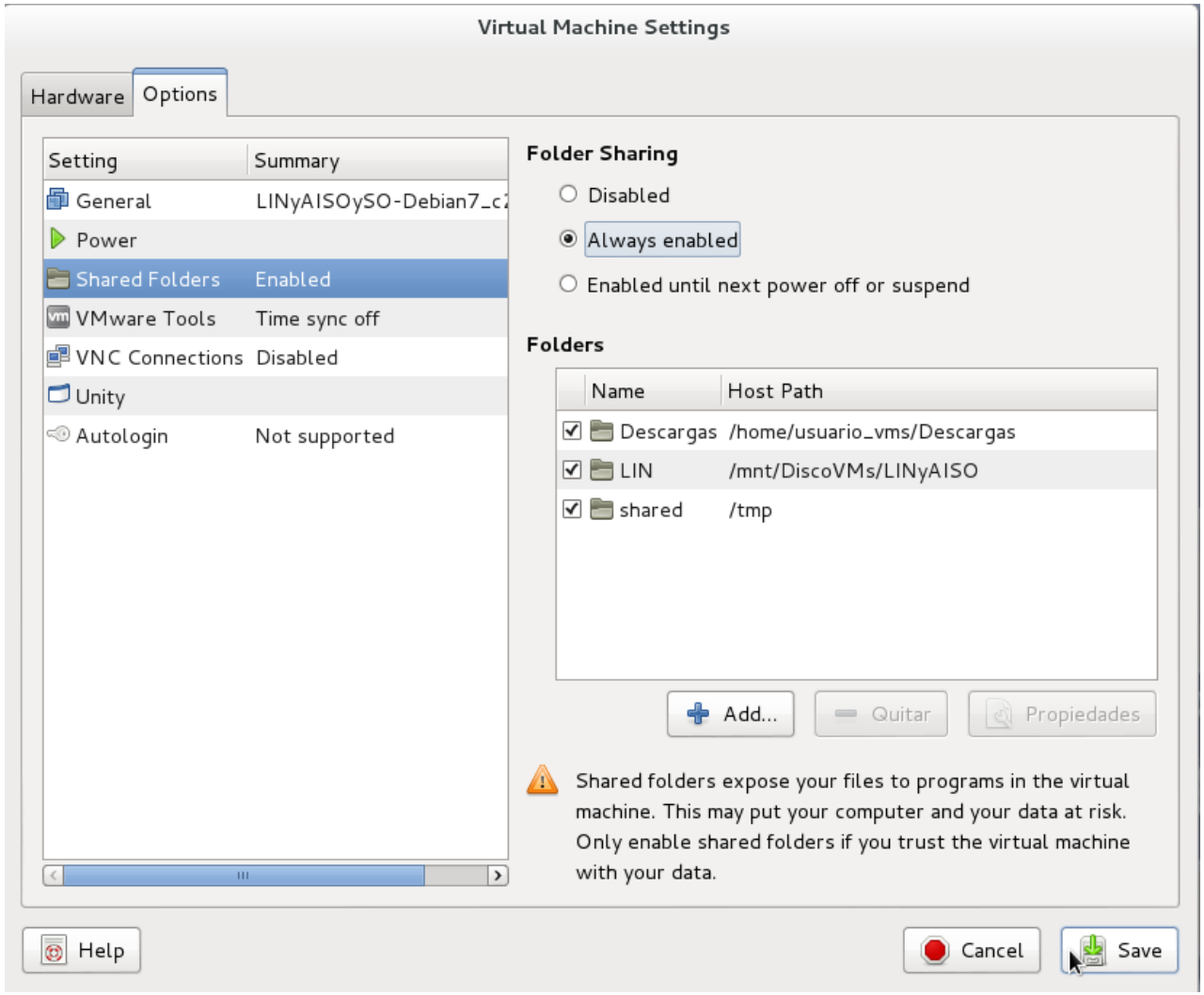
  `$ VM_LINyAISOySO-debian.sh`

- Añadir 2 carpetas compartidas a la MV asociadas a 2 directorios de Debian nativo:
  1. "LIN" → `/mnt/DiscoVMs/LINyAISO`
     - Accesible en MV vía `/mnt/hgfs/LIN`
  2. "Descargas" → `/home/usuario_vms/Descargas`
     - Accesible en MV vía `/mnt/hgfs/Descargas`

Acceso cuadro de diálogo de Carpetas compartidas

1. Virtual Machine -> Virtual Machine Settings
2. Options [Tab] -> Shared Folders [List]

Modificar configuración de la MV de Debian



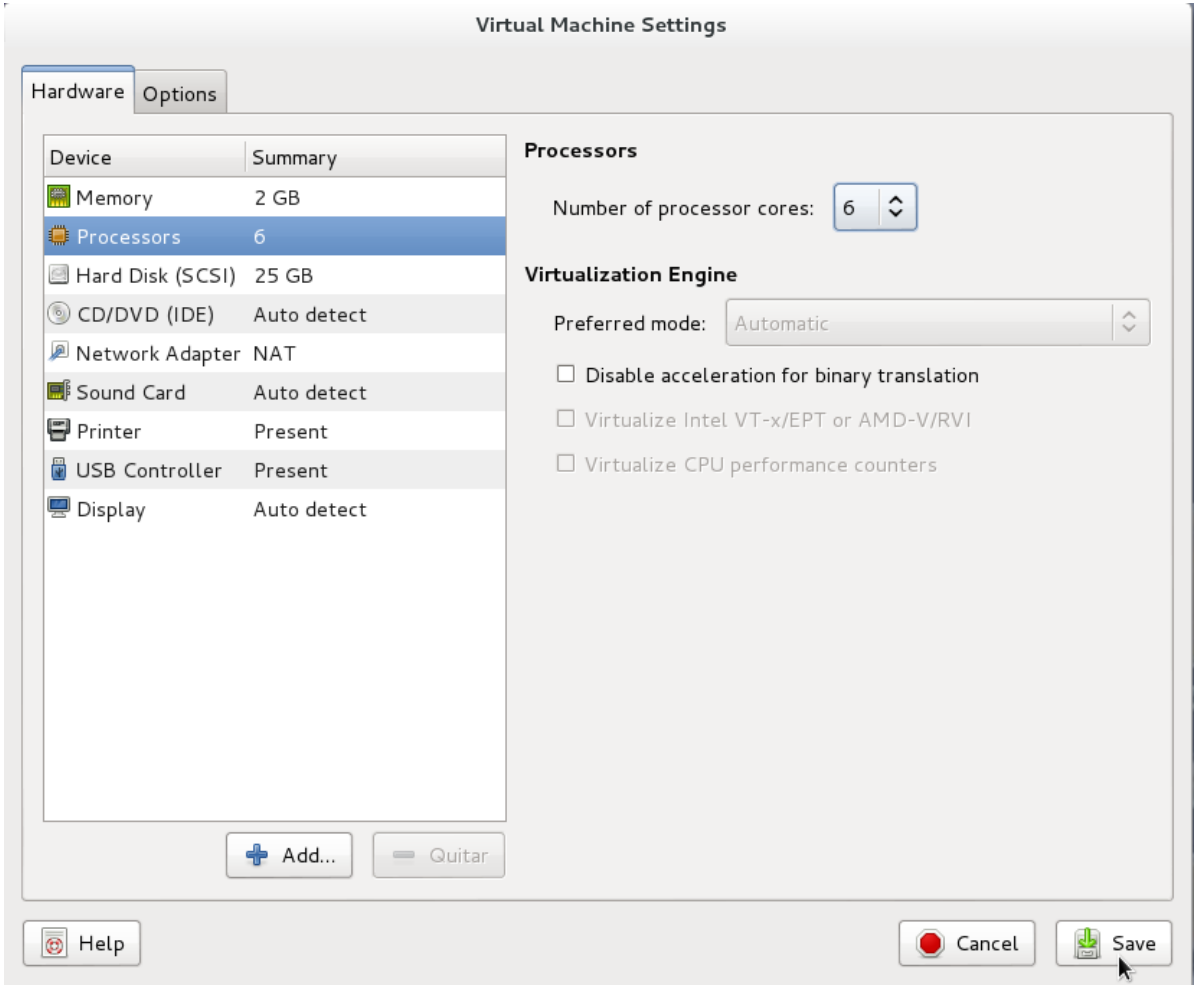
Modificar configuración de la MV de Debian

- Aumentar el número de cores de la MV a 6

Acceso configuración

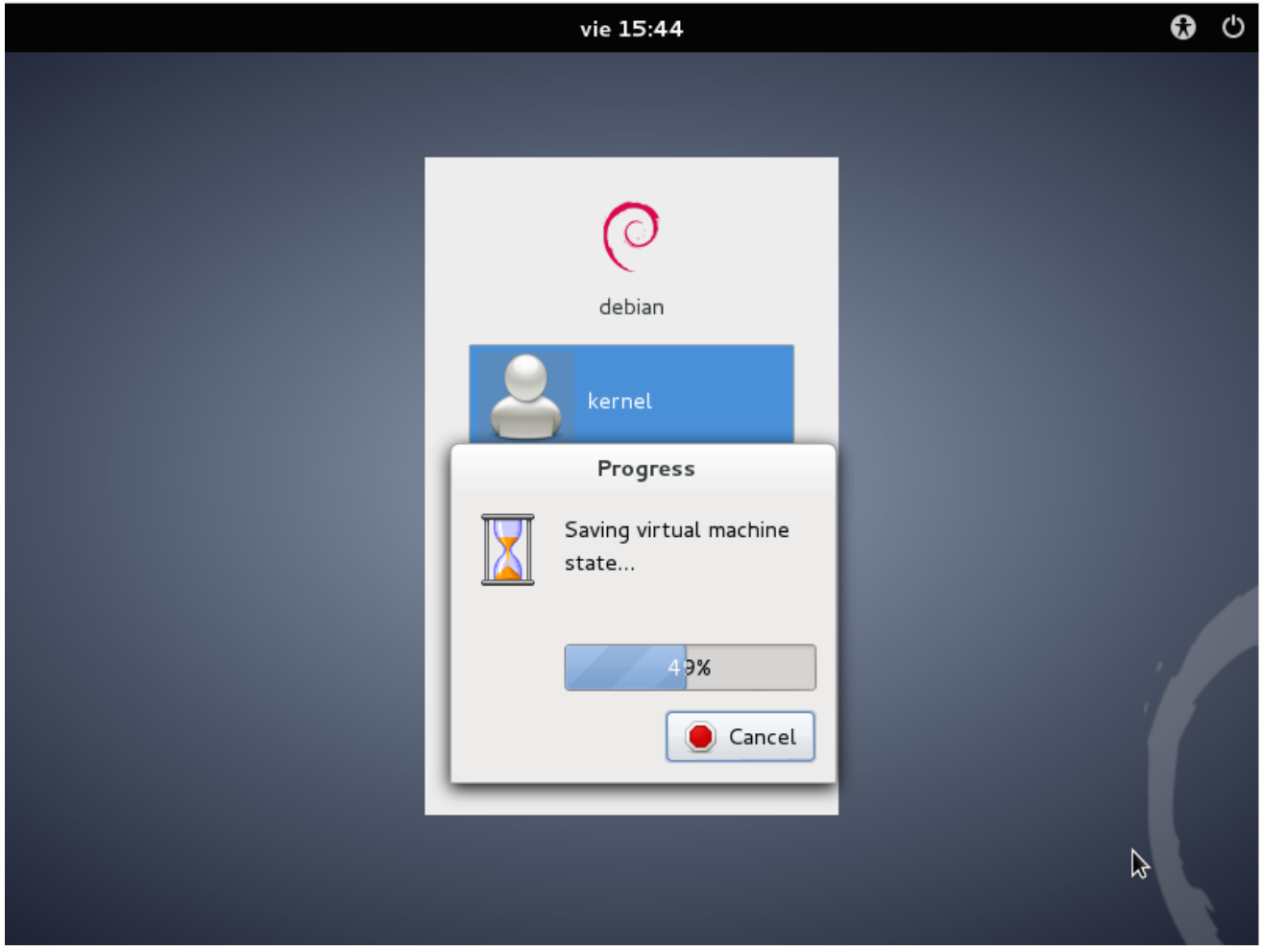
1. Virtual Machine -> Virtual Machine Settings
2. Hardware [Tab] -> Processors
3. Number of processor cores: 6

Modificar configuración de la MV de Debian



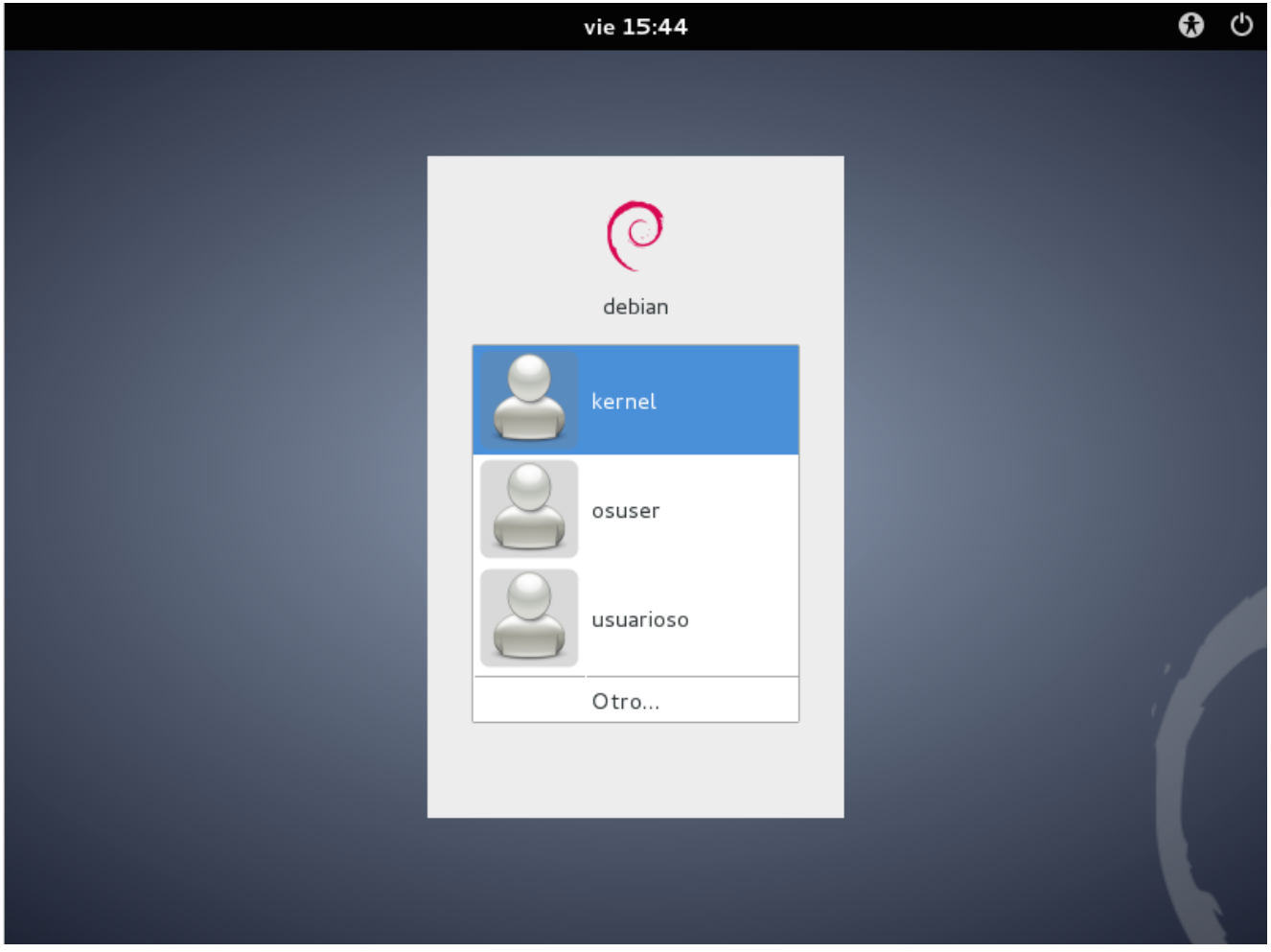
Modificar # de cores y hacer clic en "Save"

Modificar configuración de la MV de Debian



Esperar a que se apliquen los cambios

Modificar configuración de la MV de Debian



Por último, reiniciar la MV para que los cambios tengan efecto

Desplegar código de Android-x86 en MV

1. Iniciar sesión con usuario kernel y abrir una ventana de terminal
2. Descomprimir código de Android-X86 en `$HOME`:

```
$ cd
$ tar xzvf /mnt/hgfs/LIN/kitkat-src-norepo.tar.gz
```

Contenido kitkat-src-norepo.tar.gz

- Código de Android-x86 (kitkat) obtenido del repo oficial de Android x86
- Directorio .repo eliminado para ahorrar espacio

Descomprimir ficheros de ejemplo en MV

- Descomprimir ficheros de ejemplo en `$HOME`:

```
$ cd
$ zcat /mnt/hgfs/Descargas/FicherosAndroid.tar.gz | tar xzvf -
```

Contenido FicherosAndroid.tar.gz

- Fichero de configuración del kernel: `config-android-x86-lin`
  - Extraído de la MV de Android-x86 (`/proc/config.gz`)
- Dos proyectos de ejemplo para Android-x86:
  1. "Hola Mundo" en C
  2. Módulo del kernel Clipboard

Instalar Android-NDK

- Descomprimir NDK de Android en `$HOME`:

```
$ cd
$ tar xzvf /mnt/hgfs/Descargas/android-ndk-r10d.tgz
```

Android-NDK release 10

- Alternativamente, esta versión del NDK de Android puede obtenerse aquí

Liberar espacio en el disco de la MV

1. Borrar paquetes .deb de la cache del instalador

```
$ sudo rm /var/cache/apt/archives/*.deb
```

2. Borrar ficheros innecesarios en código de Android-X86

```
$ cd ~/kitkat-src
$ rm -rf external/chromium*
```

3. Verificar el espacio restante en disco con `df -h`
   - Debería haber 3.5GB libres aprox.

Configurar compilación del AOSP (1/2)

- Posicionarse en el directorio donde se encuentran las fuentes de Android-x86 y cargar comandos de compilación

```
Terminal
kernel@debian:~$ cd ~/kitkat-src
kernel@debian:~/kitkat-src$ . ./build/envsetup.sh
including device/generic/x86/vendorsetup.sh
including sdk/bash_completion/adb.bash
```

- A continuación, escoger variante (combo) del AOSP a compilar con `lunch`
  - Seleccionar Opción 5 (`android_x86-eng`) como se muestra a continuación

Configurar compilación del AOSP (2/2)

Terminal

```
kernel@debian:~/kitkat-src$ lunch

You're building on Linux

Lunch menu... pick a combo:
     1. aosp_arm-eng
     2. aosp_x86-eng
     3. aosp_mips-eng
     4. vbox_x86-eng
     5. android_x86-eng
     6. android_x86-userdebug
     7. android_x86-user

Which would you like? [aosp_arm-eng] 5

============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.4.4
TARGET_PRODUCT=android_x86
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=x86
TARGET_ARCH_VARIANT=x86
...
```

Compilación del kernel de Android-x86

- Copiar el fichero de configuración en directorio predefinido de la arquitectura

```
$ cp ~/config-android-x86-lin kernel/arch/x86/configs/
```

- Configurar+Compilar el kernel

```
$ make -j6 kernel TARGET_PRODUCT=android_x86 \
    TARGET_KERNEL_CONFIG=config-android-x86-lin
```

Advertencia

- Si la compilación se hace en la MV de Debian sin modificar, usar opcion -j3 en lugar de -j6 (en la MV original hay 2 cores)

Compilación del kernel de Android-x86 (Salida)

Terminal

```
kernel@debian:~/kitkat-src$ make -j6 kernel TARGET_PRODUCT=android_x86  \
> TARGET_KERNEL_CONFIG=config-android-x86-lin
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.4.4
TARGET_PRODUCT=android_x86
TARGET_BUILD_VARIANT=eng
....
HOST_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.14.1.lin-x86_64-with-debian-7.6
HOST_BUILD_TYPE=release
BUILD_ID=KTU84Q
OUT_DIR=out
============================================
including ./abi/cpp/Android.mk ...
including ./art/Android.mk ...
including ./bionic/Android.mk ...
bionic/libc/Android.mk:458: TARGET_CPU_VARIANT is not defined
including ./bootable/newinstaller/Android.mk ...
including ./build/libs/host/Android.mk ...
including ./build/target/board/Android.mk ...
including ./build/tools/Android.mk ...
...
```

Compilación del kernel de Android-x86

- Acabada la compilación el kernel está disponible en la siguiente ubicación:
  - `~/kit-kat/out/target/product/x86/kernel`
- *¡¡Se trata de un kernel de 32 bits!!*
  - La máquina virtual lleva Android-x86 de 32 bits

Terminal

```
kernel@debian:~/kitkat-src$ file out/target/product/x86/kernel
out/target/product/x86/kernel: Linux kernel x86 boot executable bzImage,
version 3.10.52-android-x86 (kernel@debian) #1 SMP PREEMPT Fri Jan 16 ,
RO-rootFS, swap_dev 0x3, Normal VGA
kernel@debian:~/kitkat-src$
```

- Con el kernel compilado, ya es posible compilar módulos para Android-x86

Instalación del kernel compilado

- El kernel de Android-x86 se almacena en un directorio predefinido dentro de la partición de `boot` de Android
  - `<RAIZ_BOOT>/android-4.4-r1/kernel`
- Instalar el kernel ⇒ sobreescribir versión instalada con la nuestra
- La partición de `boot` no es accesible desde Android
  - Es preciso que montemos *externamente* la partición en modo RW
  - Partición reside en el disco virtual: fichero `Android_x86_LIN.vmdk`

Advertencia

- En el laboratorio, por motivos de permisos, no es posible montar el disco virtual de la MV de Android por defecto
- Tenemos que trabajar con nuestra propia instancia de la MV
  - Exige descomprimir el tar.gz original de la MV de Android para tener acceso lectura/escritura al disco virtual

Montaje de la partición de boot

Alternativas montaje de la partición de `boot` e instalación kernel

1. Montar partición usando `vmware-mount` desde Debian nativo
2. Adjuntar el disco virtual de Android a la MV de Debian (nuestra propia instancia)

Montaje de la partición de boot (vmware-mount)

Montar partición usando vmware-mount desde Debian nativo

- Uso vmware-mount:
  - Montar Partición:

    ```
    $ vmware-mount <ruta-disco-virtual> <punto-montaje>
    ```

  - Desmontar Partición

    ```
    $ vmware-mount -d <punto-montaje>
    ```

- Necesitamos acceso como administrador (o sudo) para instalar el kernel en la partición montada con vmware-mount
  - No podemos hacerlo en el laboratorio pero sí en nuestro portátil/sobremesa

Instalación del kernel vía `vmware-mount` (1/2)

- Suponer que tar.gz con MV de Android se ha descomprimido en directorio `~/VMs` y que la partición se montará en `~/mnt`

Terminal

```
jcsaez@mi-debian:~$ mkdir mnt
jcsaez@mi-debian:~$ vmware-mount VMs/Android_x86_LIN/Android_x86_LIN.vmdk mnt
jcsaez@mi-debian:~$ find mnt/
mnt/
mnt/grub
mnt/grub/stage2_eltorito
mnt/grub/stage2
mnt/grub/stage1
mnt/grub/ntfs_stage1_5
mnt/grub/iso9660_stage1_5
mnt/grub/fat_stage1_5
mnt/grub/e2fs_stage1_5
mnt/grub/android-x86.xpm.gz
mnt/grub/menu.lst
mnt/android-4.4-r1
mnt/android-4.4-r1/system.img
mnt/android-4.4-r1/data.img
mnt/android-4.4-r1/initrd.img
mnt/android-4.4-r1/kernel
mnt/android-4.4-r1/ramdisk.img
```

Instalación del kernel vía `vmware-mount` (2/2)

- Una vez montada la partición haremos copia de seguridad del kernel existente y copiamos el nuestro en su lugar

Terminal

```
jcsaez@mi-debian:~$ cd mnt/android-4.4-r1
jcsaez@mi-debian:~/mnt/android-4.4-r1$ sudo mv kernel kernel.old
jcsaez@mi-debian:~/mnt/android-4.4-r1$ sudo cp <ruta-kernel-modificado> kernel
jcsaez@mi-debian:~/mnt/android-4.4-r1$
```

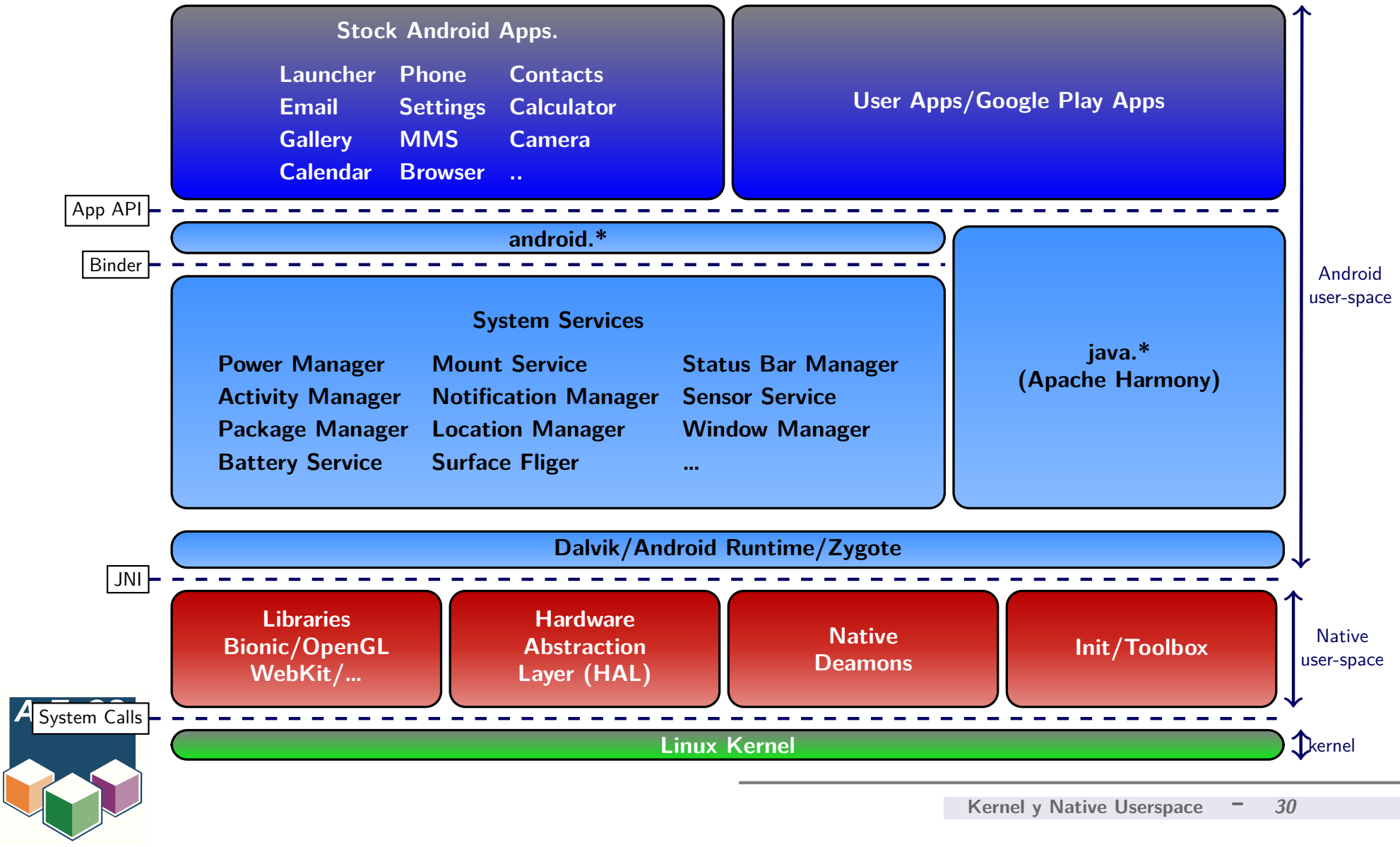
Montaje de la partición de boot (vía MV de Debian)

Pasos instalación usando segunda alternativa

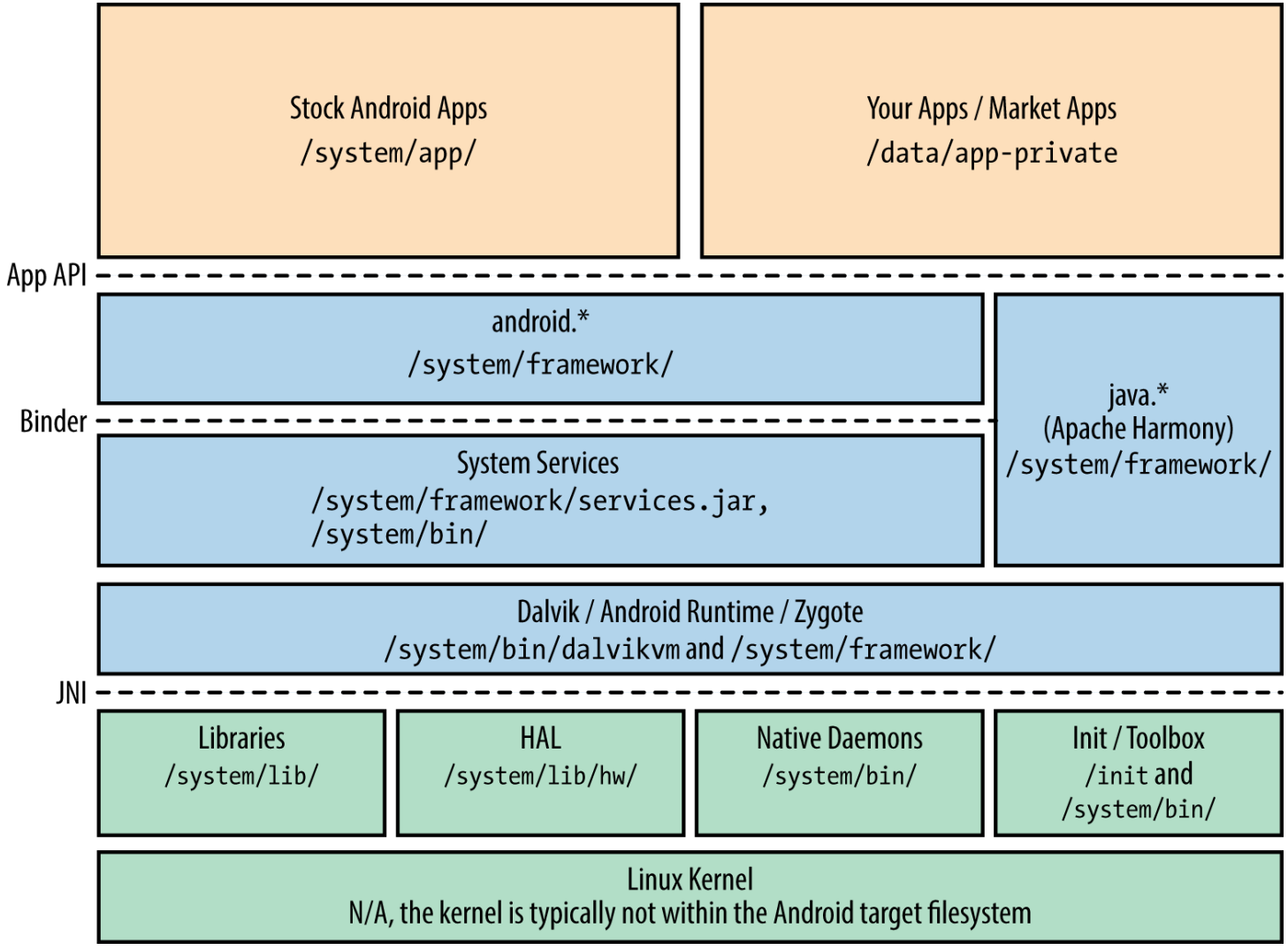
- Añadir disco extra a la MV de Debian e indicarle la ruta del disco existente `Android_x86_LIN.vmdk`
  - Implica copia del disco virtual!!
- Arrancar Debian y montar el disco
  - Aparecerá como `/dev/sdb` y partición de boot será `/dev/sdb1`
    - `sudo mount /dev/sdb1 <punto-montaje>`
  - Copiar nuestro kernel al directorio correspondiente
    - Se recomienda hacer backup del existente
  - Una vez instalado el kernel, reemplazar el disco de la MV de Android con el disco modificado que tiene nuestro kernel

Contenido

Native Userspace



Ubicación componentes software



Shell de Android

Shell (`/system/bin/sh`)

- Accesible a través de `adb shell` o aplicación Android de emulación de Terminal
- Variante de ksh de MirBSD (Licencia BSD)
  - Shell potente con extensiones similares a Bash/ksh93/zsh
  - Incluido en Android desde v4.0 (*Ice-Cream Sandwich*)
- Comandos externos básicos proporcionados por *Toolbox*
  - `cp, cat, chmod, ln, ls, mv, rm, du, rmdir, mkdir, ...`
  - `iftop, hd, cmp, getprop, setprop, getevent, sendevent, start, stop, log, ioctl, notify, schedtop, ...`
  - Comandos son enlaces simbólicos a Toolbox
    - Ej: `'/system/bin/cp' -> 'toolbox'`
- Otros comandos externos específicos de Android
  - `pm, am, netcfg, logcat, linker, run-as, ...`

Native Userspace: ADB Shell (1/2)

- **Objetivo:** iniciar un shell interactivo como root en Android desde la MV de Debian

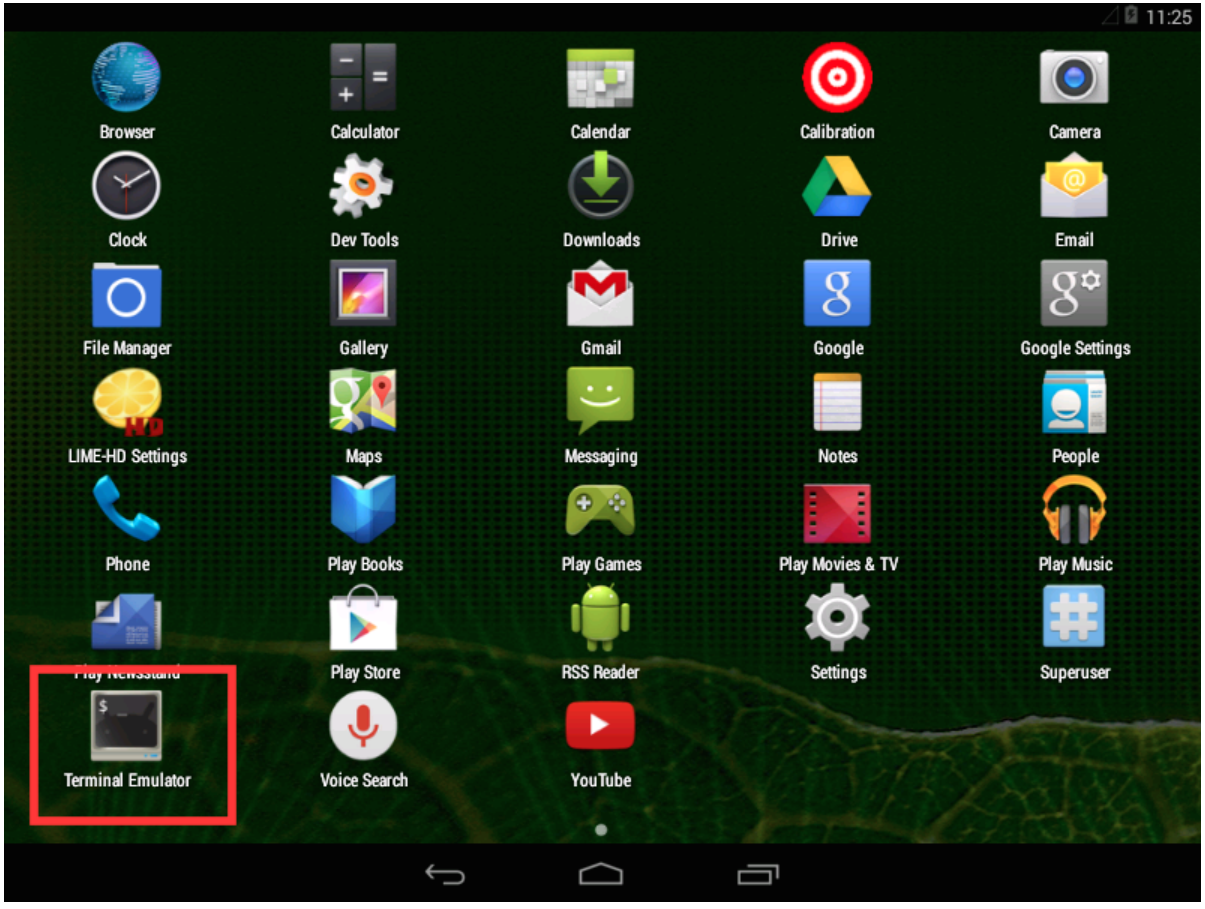
Iniciar shell interactivo como root

- Arrancar la MV de Android desde un shell de `usuario_vms`

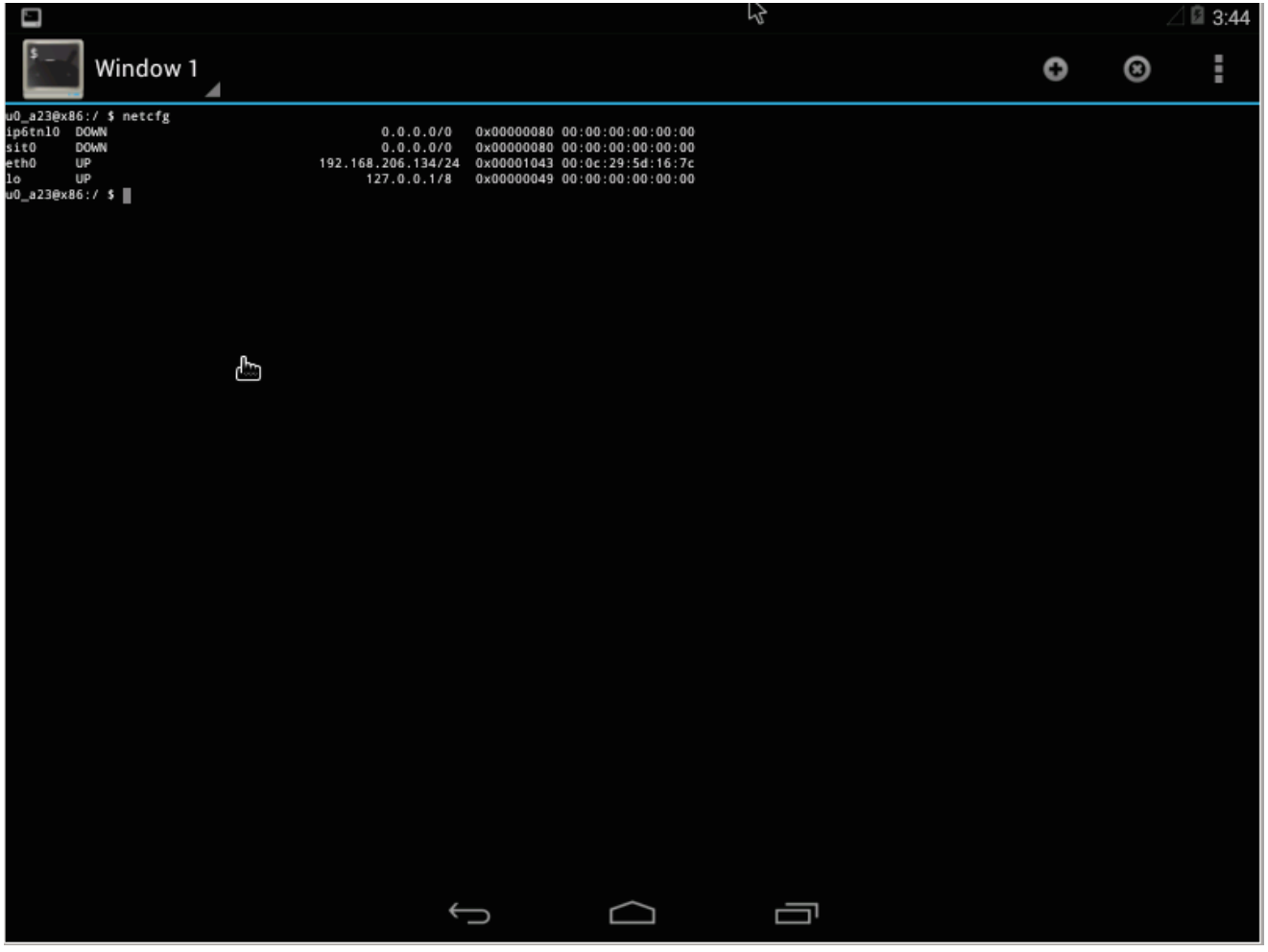
  ```
  $ VM_LINyAISO-android.sh
  ```

- Averiguar dirección IP de la MV
  - Arrancar aplicación "Terminal Emulator" desde Android Launcher
  - Ejecutar Comando `netcfg`

Averiguar Dirección IP de la MV de Android



Averiguar Dirección IP de la MV de Android



Ejecutar `netcfg`

Averiguar Dirección IP de la MV de Android

Terminal

```
$ netcfg
ip6tnl0  DOWN                                  0.0.0.0/0   0x00000080 00:00:00:00:
sit0     DOWN                                  0.0.0.0/0   0x00000080 00:00:00:00:
eth0     UP                                192.168.206.134/24  0x00001043 00:0c:29:5d:
lo       UP                                    127.0.0.1/8   0x00000049 00:00:00:00:
```

Native Userspace: ADB Shell (2/2)

Iniciar shell interactivo como root (cont.)

- Abrir terminal en MV de Debian
  - Comandos:
    - `adb connect <Dirección-IP>`: Conectarse por red a dispositivo Android vía ADB
    - `adb root`: Reiniciar proceso ADBd en Android para que se ejecute como root
    - `adb shell`: Iniciar shell interactivo

Terminal

```
kernel@debian:~$ adb connect 192.168.206.134
connected to 192.168.206.134:5555
kernel@debian:~$ adb root
restarting adbd as root
kernel@debian:~$  adb connect 192.168.206.134
connected to 192.168.206.134:5555
kernel@debian:~$ adb shell
root@x86:/ # uname -a
Linux localhost 3.10.52-android-x86+ #1 SMP PREEMPT Fri Jan 16 20:41:17
 CET 2015 i686 GNU/Linux
```

Procesos en ejecución (`ps`)

Tipos de procesos

1. Init y *kthreads*
2. Demonios de Android
   - `zygote`, `installd`, `vold`, `netd`, `rild`, `surfaceflinger`, ...
3. Aplicaciones Android (Java)
   - Hijos de `zygote`
   - Comando que figura en `ps` modificado con `prctl()`
4. Programas de línea de comando
   - Shell, Toolbox y otros

Native Userspace: (ps)

Terminal

```
root@x86:/ #   ps
USER      PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      1     0     808    512   00000000 08062dd6 S /init
root      2     0     0      0     00000000 00000000 S kthreadd
root      3     2     0      0     00000000 00000000 S ksoftirqd/0
root      5     2     0      0     00000000 00000000 S kworker/0:0H
root      7     2     0      0     00000000 00000000 S migration/0
root      8     2     0      0     00000000 00000000 S rcu_preempt
root      9     2     0      0     00000000 00000000 S rcu_bh
root      10    2     0      0     00000000 00000000 S rcu_sched
root      11    2     0      0     00000000 00000000 S watchdog/0
root      12    2     0      0     00000000 00000000 S watchdog/1
root      13    2     0      0     00000000 00000000 S migration/1
root      14    2     0      0     00000000 00000000 S ksoftirqd/1
root      15    2     0      0     00000000 00000000 S kworker/1:0
root      16    2     0      0     00000000 00000000 S kworker/1:0H
root      17    2     0      0     00000000 00000000 S khelper
root      18    2     0      0     00000000 00000000 S kworker/u4:1
root      301   2     0      0     00000000 00000000 S writeback
root      304   2     0      0     00000000 00000000 S bioset
root      305   2     0      0     00000000 00000000 S crypto
root      307   2     0      0     00000000 00000000 S kblockd
root      528   2     0      0     00000000 00000000 S ata_sff
...
```

Native Userspace: ps (Cont.)

Terminal

```
root      1208  1     1988   1488  00000000 08062dd6 S /sbin/ueventd
root      1327  1     1428   208   00000000 b76a0166 S /system/bin/powerbtnd
media_rw  1328  1     4056   448   ffffffff b7716166 S /system/bin/sdcard
root      1332  1     1572   4     00000000 0805f1bb S /sbin/healthd
system    1333  1     1468   192   00000000 4006f5a6 S /system/bin/servicemanager
root      1334  1     5480   848   ffffffff 400c0361 S /system/bin/vold
root      1335  1     11248  1380  ffffffff 4010b361 S /system/bin/netd
root      1336  1     1664   524   00000000 4008c953 S /system/bin/debuggerd
radio     1337  1     5412   836   ffffffff 4006f361 S /system/bin/rild
root      1338  1     37756  21868 ffffffff 400df09b S /system/bin/surfaceflinger
root      1339  1     921176 68932 ffffffff 40085790 S zygote
drm       1340  1     10668  2904  ffffffff 401a25a6 S /system/bin/drmserver
media     1341  1     48304  18376 ffffffff 400f85a6 S /system/bin/mediaserver
install   1342  1     1564   496   00000000 4008f166 S /system/bin/installd
keystore  1343  1     4732   1356  00000000 400d05a6 S /system/bin/keystore
root      1350  1     1552   616   00000000 400c3166 S /system/bin/sh
root      1363  1     1200   4     00000000 080e26d3 S /system/xbin/su
...
```

Native Userspace: ps (Cont.)

Terminal

```
system    1480  1339  994052 69588 ffffffff 4008709b S system_server
dhcp      1699  1     1632   500   00000000 4008a166 S /system/bin/dhcpcd
u0_a11    1714  1339  946868 58580 ffffffff 4008709b S com.android.systemui
u0_a48    1807  1339  946696 50380 ffffffff 4008709b S com.google.android.inputmeth
u0_a8     1820  1339  1086316 61868 ffffffff 4008709b S com.google.android.gms
radio     1833  1339  952528 51116 ffffffff 4008709b S com.android.phone
u0_a19    1845  1339  942100 58976 ffffffff 4008709b S com.cyanogenmod.trebuchet
u0_a63    1861  1339  931468 42248 ffffffff 4008709b S com.android.printspooler
u0_a8     1951  1339  971636 59248 ffffffff 4008709b S com.google.process.gapps
u0_a8     1981  1339  972964 59572 ffffffff 4008709b S com.google.process.location
u0_a8     2062  1339  954700 51684 ffffffff 4008709b S com.google.android.gms.weara
u0_a1     2148  1339  932124 46392 ffffffff 4008709b S com.android.providers.calend
u0_a6     2165  1339  934152 49384 ffffffff 4008709b S android.process.media
u0_a12    2270  1339  936388 45916 ffffffff 4008709b S com.android.mms
u0_a15    2317  1339  957320 49956 ffffffff 4008709b S com.android.vending
u0_a20    2366  1339  954464 55000 ffffffff 4008709b S com.google.android.googlequi
u0_a30    2395  1339  941652 45264 ffffffff 4008709b S com.android.calendar
u0_a34    2422  1339  933164 45684 ffffffff 4008709b S com.android.deskclock
u0_a38    2438  1339  941976 49772 ffffffff 4008709b S com.android.email
u0_a39    2459  1339  933144 43560 ffffffff 4008709b S com.android.exchange
...
```

Ejercicio

Ejercicio ADB

- Copiar el fichero de configuración del kernel de Android a la MV de Debian
  - Este fichero se encuentra en formato gzip dentro de Android en `/proc/config.gz`
- Para ello, realizar las siguientes acciones:
  - Iniciar un shell de root en Android vía `adb shell`
  - Copiar el fichero `/proc/config.gz` al directorio `/data/local/tmp` dentro de Android
  - Descomprimir fichero con `gunzip` en `/data/local/tmp`
  - Desde la MV de Debian (conectada por ADB a Android) transferir el fichero de configuración descomprimido en /data/local/tmp (Android) a /var/tmp (Debian)
    - Para ello usar el siguiente comando:

      ```
      adb pull <ruta-origen-android> <ruta-destino-host>
      ```

Contenido

Modificando Android

- Es posible introducir cambios en Android recompilando el AOSP
  - Para ello se pueden modificar las fuentes de algún componente
  - ... o introducir componentes nuevos
- No abordaremos compilación del AOSP en el laboratorio
  - Enormes requisitos de memoria y disco para compilación
    - Compilación de Android-x86 tardaría más de 2h30m con la configuración HW de la MV
  - Consultar link para más información requisitos compilación

Modificando Android (Cont.)

- En lugar de recompilar Android-x86 alteraremos el sistema instalado en la máquina virtual de la asignatura
  - Carga de módulos del kernel
  - Instalación de programas nativos (en C o C++)
- Es posible hacerlo porque tenemos permisos de `root`
  - En general dispositivos con Android requieren que "rooteemos" el dispositivo

Modificando Android (Cont.)

Software necesario

1. ADB
   - Lo usaremos para transferir binarios entre *host* y *target*
   - Comando instalado en MV de Debian (`/usr/bin/adb`)
2. Árbol de fuentes del kernel compilado con misma configuración que la MV de Android
   - Se requiere para compilar módulos del kernel
   - Usaremos el árbol que compilamos anteriormente
     - `~/kitkat-src/out/target/product/x86/obj/kernel`
3. Android NDK
   - Se precisa para compilar programas C contra Binder (libc)
   - Es posible construir ejecutables "funcionales" con gcc para la MV de Android-X86

Compilación y carga de módulos del kernel

- Se usa metodología similar a la empleada en las prácticas
  - *Makefile* ligeramente distinto al habitual
  - Generaremos un fichero .ko para x86 de 32 bits (Android-x86)

Makefile para `clipboard.c`

```
obj-m += clipboard.o

## Parte común módulos
ANDROIDSRC=/home/kernel/kitkat-src
KERNEL_TREE=$(ANDROIDSRC)/kernel
KERNEL_OUT=$(ANDROIDSRC)/out/target/product/x86/obj/kernel

all:
	make -C $(KERNEL_TREE) O=$(KERNEL_OUT) M=$(PWD) modules

clean:
	make -C $(KERNEL_TREE) O=$(KERNEL_OUT) M=$(PWD) clean
```

Ejemplo: `clipboard` (Compilación)

Terminal

```
kernel@debian:~$ cd ~/Projects/Clipboard
kernel@debian:~/Projects/Clipboard$ ls
clipboard.c  Makefile
kernel@debian:~/Projects/Clipboard$ make
make -C /home/kernel/kitkat-src/kernel O=/home/kernel/kitkat-src/out/targ
  M=/home/kernel/Projects/Clipboard  modules
make[1]: se ingresa al directorio `/home/kernel/kitkat-src/kernel'
  CC [M]  /home/kernel/Projects/Clipboard/clipboard.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/kernel/Projects/Clipboard/clipboard.mod.o
  LD [M]  /home/kernel/Projects/Clipboard/clipboard.ko
make[1]: se sale del directorio `/home/kernel/kitkat-src/kernel'
kernel@debian:~/Projects/Clipboard$ file clipboard.ko
clipboard.ko: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV),
 BuildID[sha1]=0x99ecc6cc44e851d5842e363735603414d9d5d8bc, not stripped
kernel@debian:~/Projects/Clipboard$
```

Ejemplo: `clipboard` (Transferencia a la MV)

- Asegurarse de que ADBd está corriendo como root en MV y transferir fichero con `adb push`

  `$ adb push <ruta-fichero-origen-host> <ruta-destino-android>`

Terminal

```
kernel@debian:~/Projects/Clipboard$ adb connect 192.168.206.134
connected to 192.168.206.134:5555
kernel@debian:~/Projects/Clipboard$ adb root
restarting adbd as root
kernel@debian:~/Projects/Clipboard$ adb connect 192.168.206.134
connected to 192.168.206.134:5555
kernel@debian:~/Projects/Clipboard$ adb push clipboard.ko /data/local/tmp
1403 KB/s (3823 bytes in 0.002s)
kernel@debian:~/Projects/Clipboard$
```

ArTeCS

Ejemplo: `clipboard` (Carga + Test)

Terminal

```
kernel@debian:~/Projects/Clipboard$ adb shell
root@x86:/ # cd /data/local/tmp
root@x86:/data/local/tmp # insmod clipboard.ko
root@x86:/data/local/tmp # lsmod | head
clipboard 1306 0 - Live 0x00000000 (O)
binfmt_misc 4927 1 - Live 0x00000000
vivi 11158 0 - Live 0x00000000
videobuf2_vmalloc 2588 1 vivi, Live 0x00000000
videobuf2_memops 2142 1 videobuf2_vmalloc, Live 0x00000000
videobuf2_core 23165 1 vivi, Live 0x00000000
mperf 1039 0 - Live 0x00000000
mac_hid 2709 0 - Live 0x00000000
uvesafb 19713 2 - Live 0x00000000
cn 3239 2 uvesafb, Live 0x00000000
root@x86:/data/local/tmp # cd /
root@x86:/ # echo "un modulo del kernel en Android" > /proc/clipboard
root@x86:/  # cat /proc/clipboard
un modulo del kernel en Android
root@x86:/ #
```

Compilación y ejecución programas en C

- En general, programas ejecutables de Linux no funcionan en Android
  - GNU Libc no está disponible en Android
  - Ejecutables de Android enlazados contra Bionic (libc Android)

2 alternativas disponibles

1. Generación de ejecutable estático desde Linux con GCC
   - Suele requerir uso de compilador cruzado
2. Construir ejecutable con Android NDK

Compilando programas para Android con GCC

- La máquina virtual de Android-x86 lleva un SO de 32 bits
  - Ejecutable: binario de 32 bits para x86
  - Compilador GCC instalado en MV Debian preparado para generar estos ejecutables
- Opciones GCC:
  - Compilación: `-m32`
    - Generar binario de 32 bits
  - Enlazado: `-static`
    - Generar ejecutable estático

Ejemplo

```
$ gcc main.c -m32 -static -o main
```

Ejemplo: HelloAndroid (GCC)

Terminal

```
kernel@debian:~/Projects/HelloAndroid$ gcc jni/hello.c -m32 -static -o hello-androi
kernel@debian:~/Projects/HelloAndroid$ file hello-android
hello-android: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV)
statically linked, for GNU/Linux 2.6.26, BuildID[sha1]=0x17ea0df20e4052ef
f6a86cfe633c67efc24901c7, not stripped
kernel@debian:~/Projects/HelloAndroid$ adb push hello-android /data/local/tmp
2328 KB/s (591061 bytes in 0.002s)
kernel@debian:~/Projects/HelloAndroid$ adb shell
root@x86:/ # cd /data/local/tmp/
root@x86:/data/local/tmp # ./hello-android
/system/bin/sh: ./hello-android: can't execute: Permission denied
126|root@x86:/data/local/tmp # stat ./hello-android
  File: './hello-android'
  Size: 591061      Blocks: 1168       IO Block: 4096   regular file
Device: 701h/1793d  Inode: 8197        Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-01-17 19:00:14.000000000
Modify: 2016-01-17 19:00:14.000000000
Change: 2016-01-17 19:07:13.000000000
root@x86:/data/local/tmp # chmod 755 hello-android
root@x86:/data/local/tmp # ./hello-android
Hello Android
root@x86:/data/local/tmp #
```

Ejemplo: HelloAndroid (Cont.)

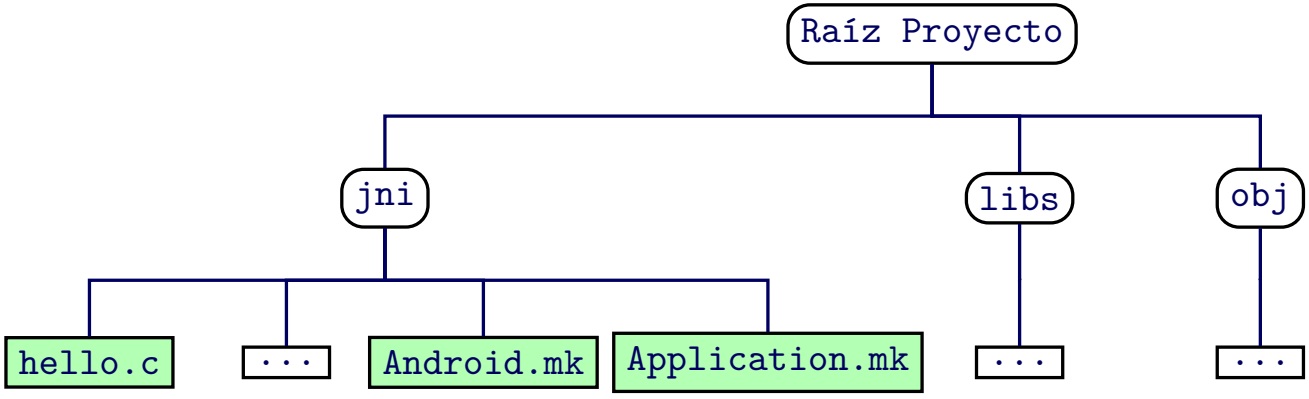
- Alternativamente, una vez instalado, se puede iniciar su ejecución desde el *host* pasando la ruta del ejecutable como argumento del comando `adb shell`

Terminal

```
kernel@debian:~/Projects/HelloAndroid$ adb shell /data/local/tmp/hello-android
Hello Android
kernel@debian:~/Projects/HelloAndroid$
```

Proyecto NDK (Programa en C)

- Construir ejecutable exige construir proyecto NDK con cierta estructura



3 directorios

- `jni`: Almacena fuentes y "Makefiles"
- `libs`: donde se almacena ejecutable o librerías generadas
- `obj`: Almacena ficheros intermedios de la compilación (se crea si no existe)

Control de la compilación

- Para construir programas en C es preciso incluir 2 ficheros:
  1. `Android.mk`: Makefile con formato del AOSP
  2. `Application.mk`: Define variable `APP_ABI` que indica las arquitecturas para las que se compila el proyecto
     - `APP_ABI := x86`

`Android.mk` (Ejemplo Hello)

```
LOCAL_PATH := $(my-dir)
include $(CLEAR_VARS)

LOCAL_SRC_FILES:= hello.c  ## Lista de ficheros .c del proyecto

LOCAL_SHARED_LIBRARIES :=

LOCAL_CFLAGS :=

LOCAL_MODULE := hello      ## Nombre del ejecutable

include $(BUILD_EXECUTABLE) ## Incluye reglas para compilar ejecutable
```

Control de la compilación (Cont.)

- Compilación gestionada mediante `ndk-build`
  - Se encuentra en la raíz del NDK

Acciones básicas `ndk-build`

- Compilar proyecto:
  - Ejecutar "`ndk-build`" desde raíz del proyecto
- *Limpiar* proyecto:
  1. Ejecutar "`ndk-build clean`" desde raíz del proyecto
  2. Eliminar directorio `obj` manualmente

Ejemplo: HelloAndroid

Terminal

```
kernel@debian:~/Projects/HelloAndroid$ ~/android-ndk-r10d/ndk-build
[x86] Compile        : hello <= hello.c
[x86] Executable     : hello
[x86] Install        : hello => libs/x86/hello
kernel@debian:~/Projects/HelloAndroid$ adb push libs/x86/hello /data/local/tmp
2328 KB/s (5268 bytes in 0.002s)
kernel@debian:~/Projects/HelloAndroid$ adb shell
root@x86:/ # cd /data/local/tmp/
root@x86:/data/local/tmp # ./hello
/system/bin/sh: ./hello: can't execute: Permission denied
126|root@x86:/data/local/tmp # stat hello
  File: 'hello'
  Size: 5268            Blocks: 16         IO Block: 4096   regular file
Device: 701h/1793d      Inode: 8196        Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2015-01-16 19:42:32.000000000
Modify: 2015-01-16 19:42:32.000000000
Change: 2015-01-16 19:43:16.000000000
root@x86:/data/local/tmp # chmod 755 hello
root@x86:/data/local/tmp # ./hello
Hello Android
root@x86:/data/local/tmp
```

Referencias

Embedded Android

- Cap. 3 *"AOSP Jump-Start"*
- Cap. 4 *"The Build System"*
- Cap. 6 *"Native User-Space"*

Enlaces

- Compilación del kernel en Android-x86
- Anatomía de la instalación de Android-x86
- Estructura proyecto NDK
- Estructura fichero Android.mk
- Documentación de ADB

Licencia

LIN - Kernel y Native Userspace
Versión 0.2

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