Curso LINUX

AREA 1: Compilación cruzada

Indice

- Introducción
- Virtualización
- Compilación cruzada
- Conclusiones

Introducción

- Compilación cruzada:
- La compilación de código bajo una arquitectura para generar ejecutables para otra arquitectura
- Sistema **huesped** : sistema en el que se realiza la compilación
- Sistema **objetivo** : sistema para el que se realiza la compilación

Virtualización

- Abstracción de los recursos de una máquina en concreto
- Se crean maquinas virtuales con dispositivos genéricos
- El código se ejecuta en una máquina independiente
- Es un “envoltorio” que genera una máquina completa dentro de otra

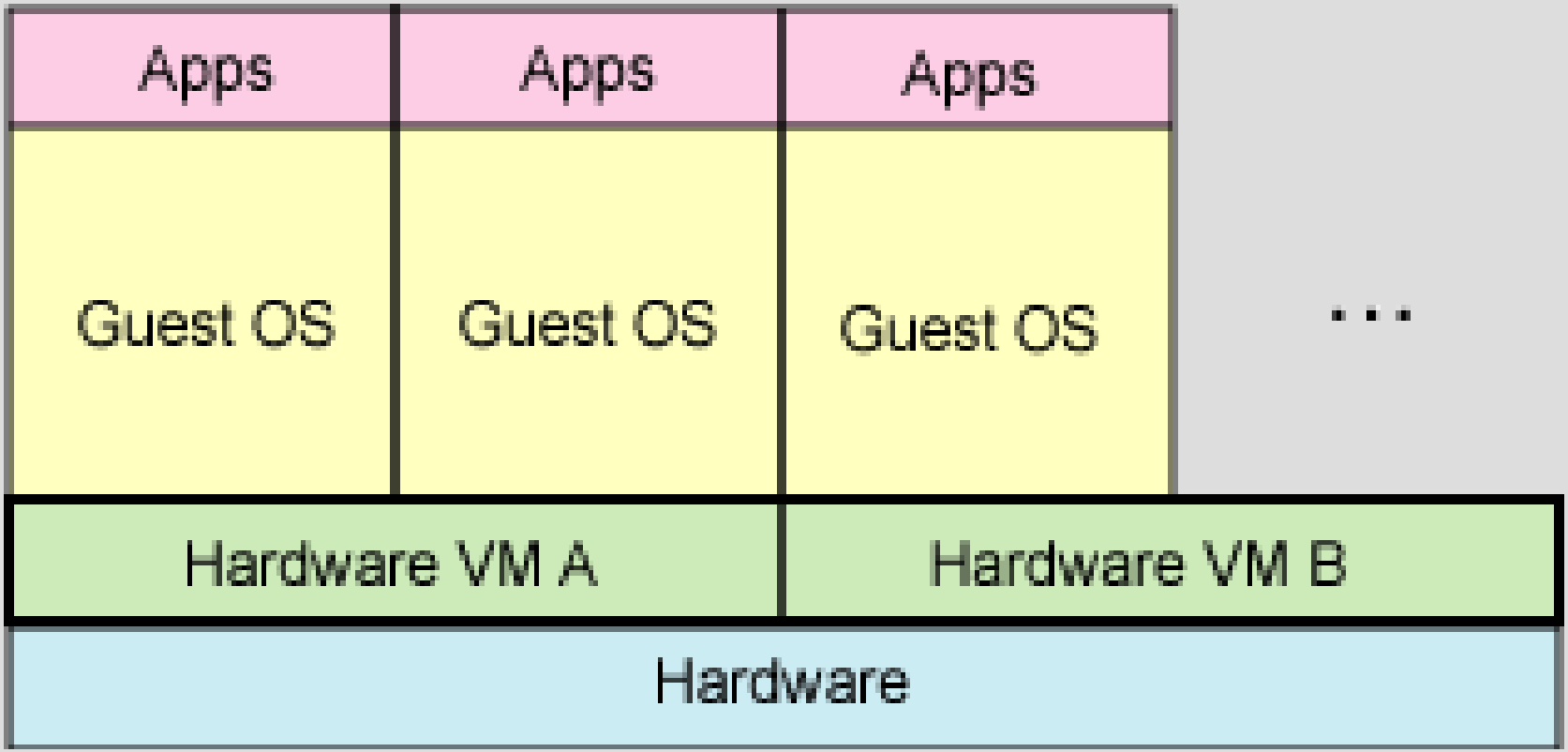
Virtualizacion (tipos)

- Emulación
  - qemu
- Virtualización completa
  - Vmware, Virtual box, ...
- Paravirtualización
  - Paralells, Vmware ESX Server, ..
- A nivel de Sistema operativo
  -

Emulación

- La máquina virtual simula un hardware completo
- Sistema operativo “guest” sin modificar para una CPU completamente diferente.
- Utilizado para permitir la creación de software para nuevos procesadores antes de que estuvieran físicamente disponibles.
- Bochs, PearPC, **Qemu**, Hercules.
- Variedad de técnicas, state machines, recopilación dinámica.

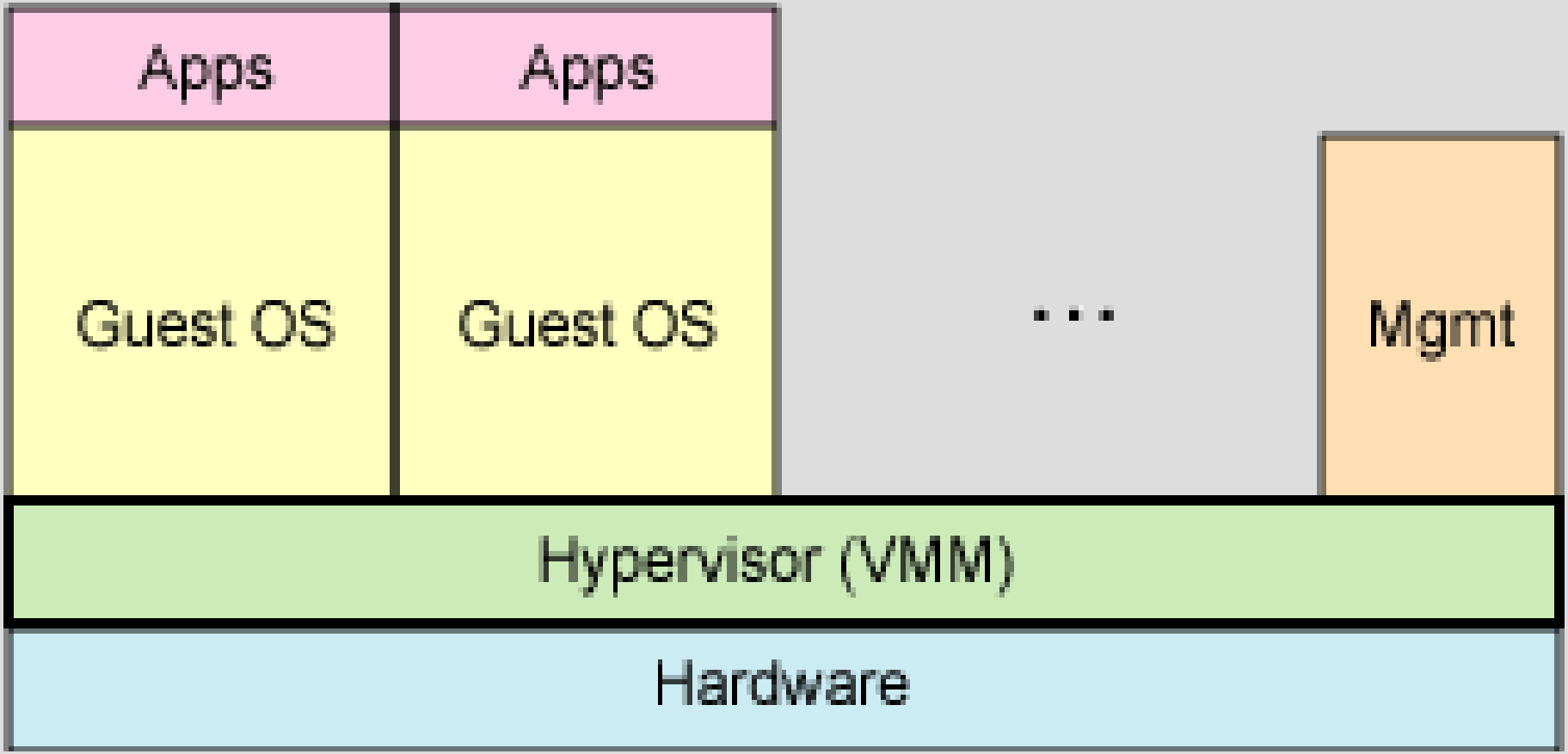
Emulación


Apps
Apps
Apps
Guest OS
Guest OS
Guest OS
. . .
Hardware VM A
Hardware VM B
Hardware

Virtualización completa

- Permiten la ejecución de sistemas huespedes sin modificar
- Utilizan un Virtual Machine Monitor o hypervisor para compartir el hardware real
- Rendimiento mayor que la emulación y menor que la nativa
- Mejoras en x86 Intel VT y AMD-V
- La virtualización no emula todo el hardware

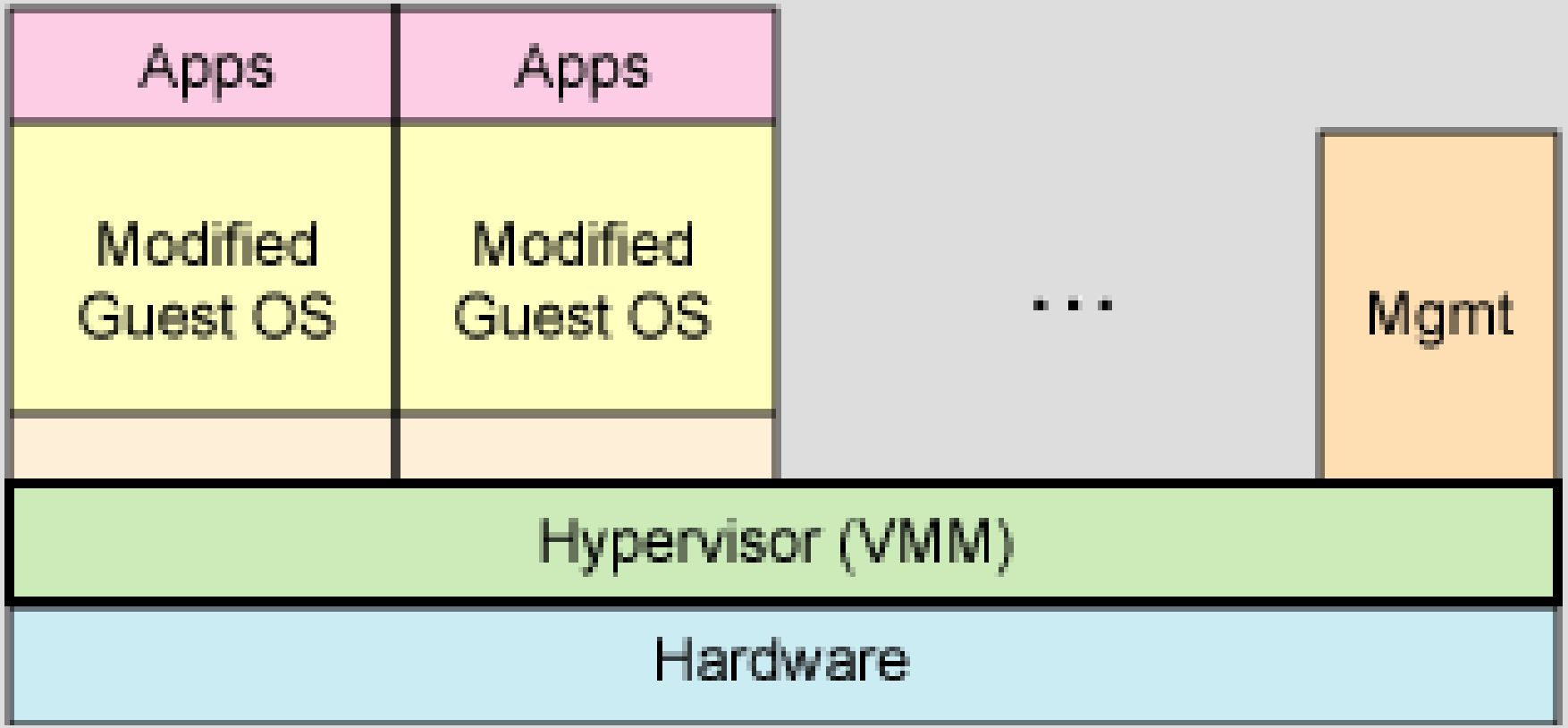
Virtualización completa



Paravirtualización

- Mejora de rendimiento sobre la virtualización completa
- Requieren un sistema huesped modificado
- El hypervisor no requiere monitorizar todas las instrucciones, sino que los huespedes colaboran en esa tarea
- Ejemplo : xen

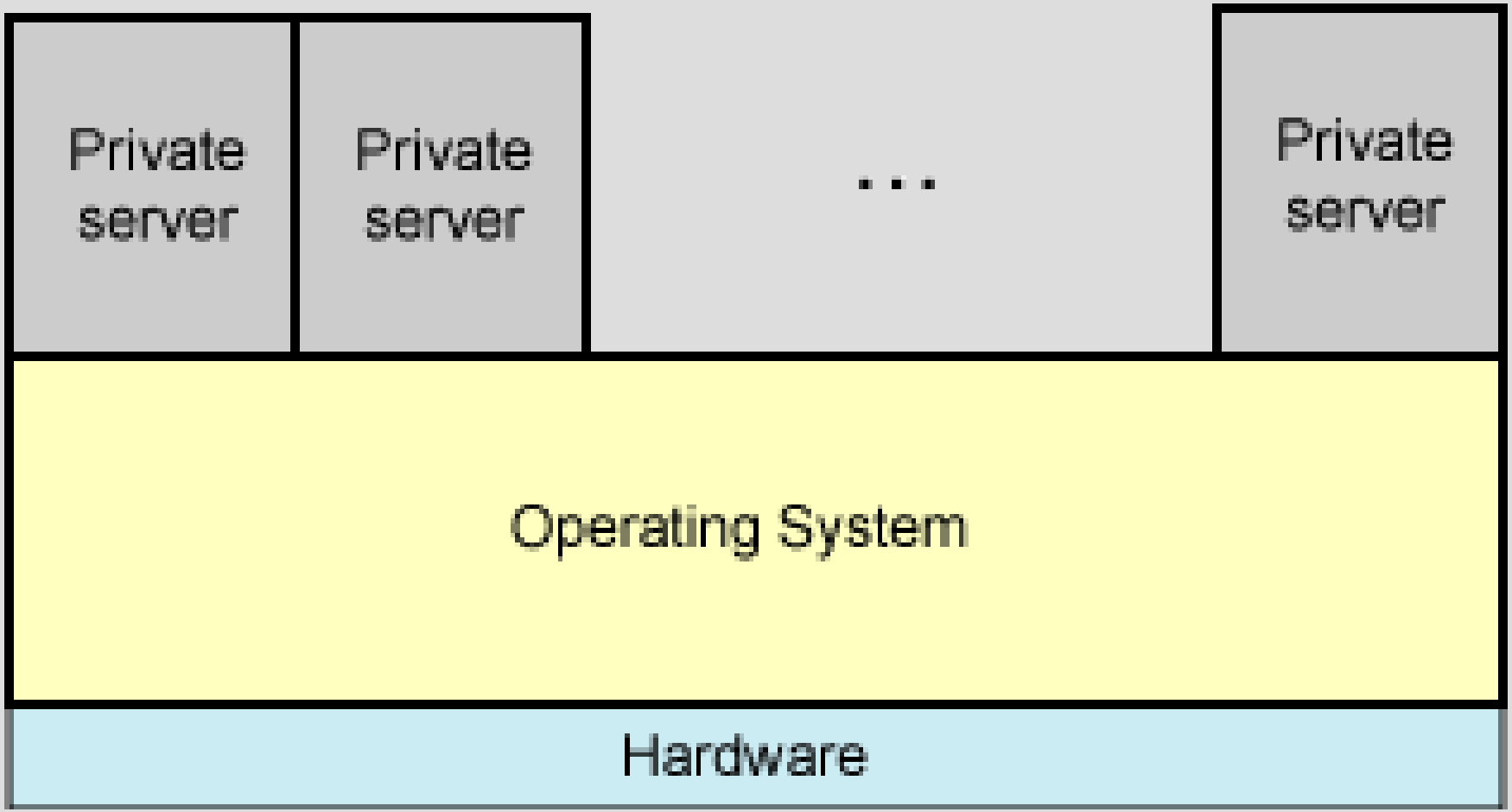
Paravirtualización



Virtualización SO

- Permite múltiples servidores virtuales aislados y seguros correr en un solo servidor físico
- El entorno del sistema operativo huesped comparte el mismo sistema operativo que el del sistema real
- Las aplicaciones del huesped lo ven como un sistema autónomo.
- Ejemplos: Linux-VServer, Solaris containers

Virtualización - SO



Virtualización - usos

- Ejecución de multiples S.O en la misma máquina (virtualizacion completa)
- Ejecución de código de arquitecturas distintas (emulación)
- Control de ejecución y posibilidad de realizar cluster a nivel de máquinas virtuales
- Tolerancia a fallos y copia instantanea de imágenes de máquinas virtuales

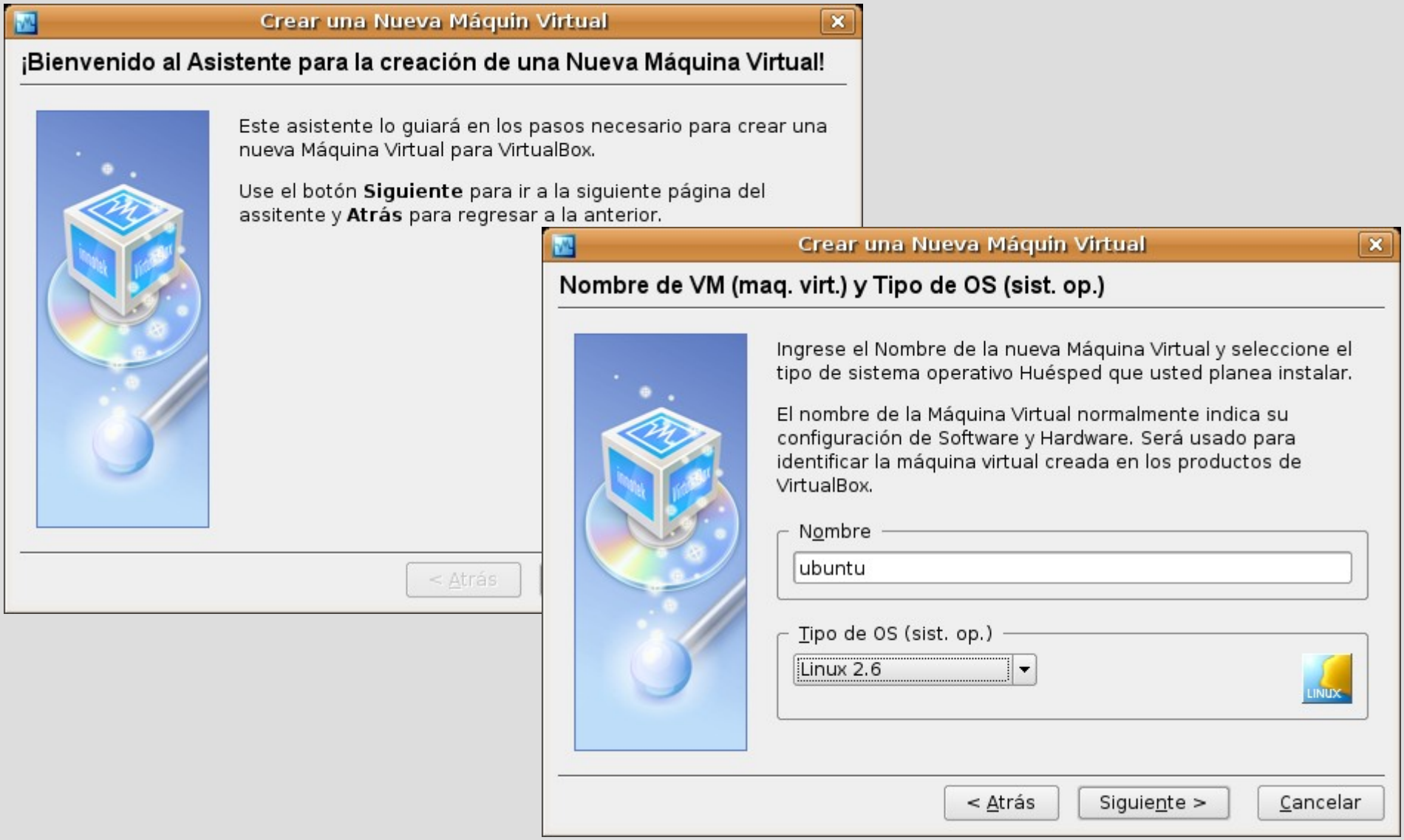
Virtualización - problemas

- Compatibilidad (emulación)
- Hardware limitado
- Rendimiento menor que el nativo
- No resuelve todos los problemas (ahora es moda)

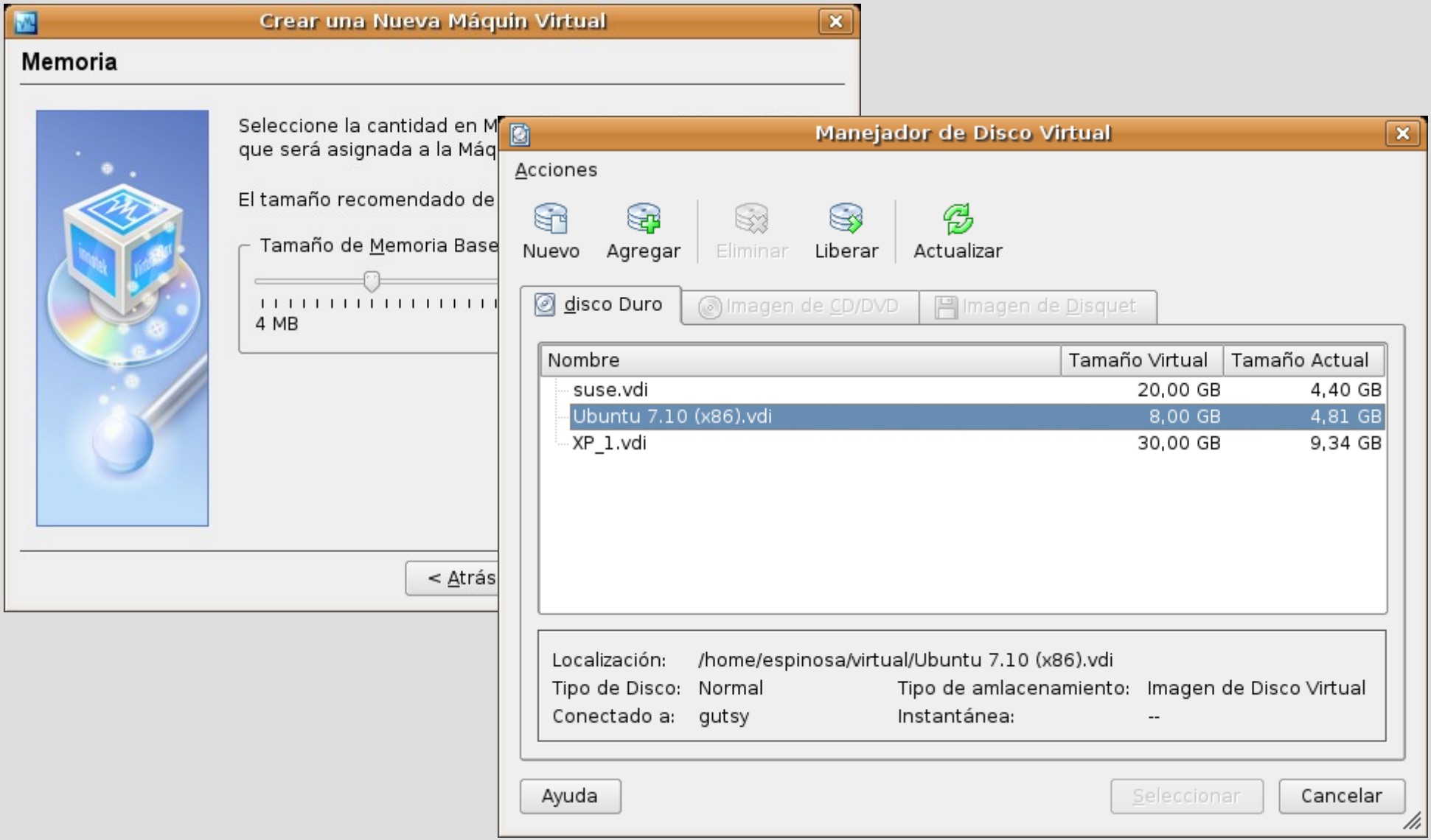
Mejor verlo

- Comprobad que teneis instalado virtualbox
- Copiad el archivo del cd
  - `Ubuntu 7.10 (x86).vdi`
- Ejecutar el comando vbox

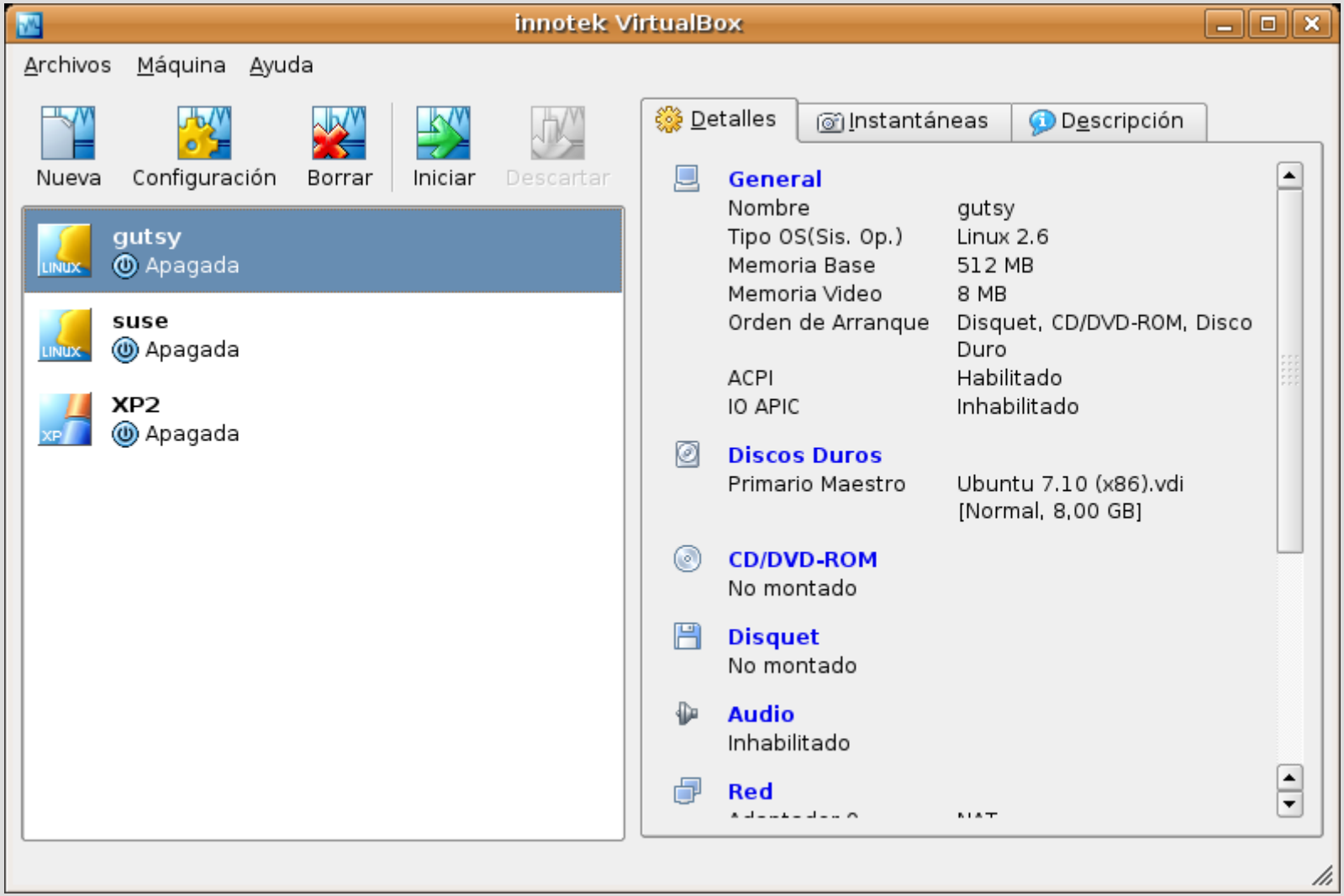
Virtual Box


Crear una Nueva Máquin Virtual
¡Bienvenido al Asistente para la creación de una Nueva Máquina Virtual!
Este asistente lo guiará en los pasos necesario para crear una nueva Máquina Virtual para VirtualBox.
Use el botón Siguiente para ir a la siguiente página del assitente y Atrás para regresar a la anterior.
< Atrás
Crear una Nueva Máquin Virtual
Nombre de VM (maq. virt.) y Tipo de OS (sist. op.)
Ingrese el Nombre de la nueva Máquina Virtual y seleccione el tipo de sistema operativo Huésped que usted planea instalar.
El nombre de la Máquina Virtual normalmente indica su configuración de Software y Hardware. Será usado para identificar la máquina virtual creada en los productos de VirtualBox.
Nombre
ubuntu
Tipo de OS (sist. op.)
Linux 2.6
LINUX
< Atrás
Siguiente >
Cancelar

Virtual Box (2)



Virtual Box (3)



Entrar en el sistema

- usuario: ubuntu
- password: page

Compilación cruzada

- Las compilaciones para una plataforma no funcionan en otra
- Ejemplo:
  - Compilar el hola mundo del día 1 en suse (x86_64)
  - Ejecutar `file hello`
  - Compilar el mismo hola mundo en la máquina virtual (i386)
  - Ejecutar `file hello`

Como funciona

- Cada uno de los pasos de generación debe estar adaptado a la máquina objetivo:
  - compilado
  - ensablado
  - enlazado
  - librerías estandar (libc)
- Podemos construir cualquier compilador cruzado, ya que disponemos de las fuentes de gcc

El método “artesanal”

- Para poder compilar en otra plataforma hay que constuir:
  - binutils
  - gcc
  - glibc / newlibc / ulibc ..
- Se necesitan los archivos de cabecera .h
- Opcional: gdb

1. Preparación

- Ubicación y coexistencia
  - Determinar el prefijo
- Crear el entorno de compilación
  - mkdir crossbuild
  - Conseguir las últimas fuentes de gcc, binutils, gdb, glibc y el kernel y descomprimirlas

Preparación : fuentes

- ftp://ftp.uvsq.fr/pub/gcc/releases/gcc-4.2.3/gcc-4.2.3.tar.bz2
- http://ftp.gnu.org/gnu/binutils/binutils-2.18.tar.bz2
- http://ftp.gnu.org/gnu/glibc/glibc-2.7.tar.bz2
- http://ftp.gnu.org/gnu/gdb/gdb-6.7.tar.bz2
- ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.24.tar.bz2

Preparación - Fin

- Creamos directorios
  - `mkdir build-binutils`
  - `mkdir build-gcc`
  - `mkdir build-glibc`
  - `mkdir build-gdb`
- Si queremos crear para más de una plataforma deberíamos crear directorios separados

2. Configuración

```
export TARGET=i386-elf
export PREFIX=/usr/local/crossgcc
export TARGET_PREFIX=$PREFIX/$TARGET
export PATH=$PATH:$PREFIX/bin
```

- Obtener los headers

```
cd linux-2.6.24
make ARCH=i386 CROSS_COMPILE=i386-linux- menuconfig
```

- Y copiarlos

```
mkdir -p $TARGET_PREFIX/include
cp -r include/linux $TARGET_PREFIX/include
cp -r include/asm-x86 $TARGET_PREFIX/include/asm
cp -r include/asm-generic $TARGET_PREFIX/include/
```

3. Construir utilidades

- Construir bintutils

```
cd build-binutils
../binutils-2.18/configure --target=i386-elf --
prefix=$PREFIX --disable-nls -v
make all
make install
```

Revisar el contenido de $TARGET_PREFIX/bin para ver el resultado

NOTA: probablemente necesite texinfo y alguna utilidad extra

Construir gcc

- stage 1

```
cd build-gcc
../gcc-4.2.3/configure --target=$TARGET --prefix=$PREFIX \
                       --without-headers --with-newlib -v
make all-gcc
make install-gcc
```

NOTA: requiere bison y alguna otra utilidad instalada

Construir glibc

```
cd ../build-gcc
CC=${TARGET}-gcc ../glibc-2.7/configure --target=$TARGET \
  --prefix=$PREFIX --with-headers=${TARGET_PREFIX}/include
make all

make install_root=${TARGET_PREFIX} prefix="" install
```

NOTA. es posible que en este momento eche de menos una librería crt0.o

Construir gcc final

- Stage 2 (ya con libc instalado)

```
cd build-gcc
rm -rf *
../gcc-4.2.3/configure --enable-languages=c \
--target=$TARGET --prefix=$PREFIX
make all
make install
```

Usar el nuevo compilador

- `i386-linux-gcc hello.c -o hello`
- Si usamos make
  - `make CC=i386-linux-gcc`
- Usando configure
  - `CC=i386-linux-gcc ./configure`
  - `./configure –host=i386-linux`
- Para las librerías habría que indicar target al igual que anteriormente

Toolchain

- Herramientas necesarias para compilar un programa para una arquitectura distinta:
  - make
  - binutils (linker, assembler, etc.)
  - gcc
  - glibc
  - Cabeceras del sistema
  - gdb (opcional)

Toolchains y más

- Existen completos sdk's para entornos embebidos
  - Buildroot (ucLib,busybox)
  - Scratchbox (maemo)
- Disponen no solo de toolbox, sino de muchas herramientas adicionales
  - emuladores
  - sistemas de paquetes
  - ...

Ejemplo scratchbox

- En la máquina ubuntu teneis instalado unscratchbox
  - `sudo /scratchbox/sbin/sbox_ctl restart`
  - `/scratchbox/login`
- Entraremos en un entorno completo con las herramientas adecuadas para
  - SDK_ARMEL
  - SDK_X86

Ejemplo scratchbox (2)

- Compilar el hola mundo...
- `file hello`

Referencias

- linuxemb.wikidot.com/tesis-c3
- https://www6.software.ibm.com/developerworks/education/l-cross/index.html
- www.scratchbox.org/
- buildroot.uclibc.org/
- www.eslomas.com/index.php/archives/2007/01/11/a-vueltas-con-la-virtualizacion