

UNIVERSIDAD AUTÓNOMA DE YUCATÁN

FACULTAD DE MATEMÁTICAS

ANEXO DE TESIS DE ALEX ANTONIO TURRIZA SUÁREZ

Configuración de una Máquina Virtual de arquitectura ARM bajo GNU/Linux

Autor:

Alex Antonio TURRIZA SUÁREZ

26 de octubre de 2016

Índice

1. Introducción	2
2. Software a utilizar	3
2.1. QEMU	3
2.2. Debootstrap	3
3. Descarga e instalación	4
3.1. Configuración del sistema	5
3.2. Ejemplo: compilando un programa en C++	8

1. Introducción

Los equipos de cómputo actuales se pueden categorizar de acuerdo a su arquitectura. Se define a la *arquitectura de una computadora* como la estructura operacional de este dispositivo. Incluye los formatos de información, conjunto de instrucciones y técnicas de direccionamiento de memoria [Mano, M. MORRIS (1994). *Arquitectura de Computadoras*. Pearson Educación].

Entre las arquitecturas más conocidas están la x86, la x86_64 o AMD64, ARM, PowerPC, entre otras. Un programa codificado para cierta arquitectura no podrá ser funcional en otra, salvo que la misma explícitamente indique la compatibilidad (por ejemplo, el caso de x86_64, que permite la ejecución de programas codificados en x86, por ser sólo una extensión de este último).

Cuando en un determinado proyecto coexisten dispositivos de cómputo de distinta arquitectura, es lógico pensar que alguna puede ser no muy cómoda para programar grandes cantidades de líneas de código, o bien, requiere de mucho tiempo al momento de compilar y obtener un ejecutable. En estos casos, toma sentido el ser capaz de compilar todo el código de determinado equipo. Esto es posible mediante la emulación de un sistema con determinada arquitectura, en otro equipo no necesariamente con la misma configuración.

En este anexo, se mostrarán los pasos a realizar para emular un dispositivo BeagleBone (con sistema operativo GNU/Linux Debian de arquitectura ARM) en una computadora convencional (con sistema operativo Ubuntu de arquitectura x86_64).

2. Software a utilizar

2.1. QEMU



Figura 1: Logotipo del software QEMU.

QEMU es un emulador y virtualizador de máquinas, de código abierto. Cuando es usado como emulador, puede correr sistemas operativos y programas hechos para determinada arquitectura. Tiene un buen rendimiento por usar lo que llama *traducción dinámica*.

2.2. Debootstrap

Debootstrap es una herramienta cuyo propósito es la instalación de un sistema basado en Debian en un directorio determinado de un sistema ya instalado. Posee la capacidad de instalar un sistema de diferente arquitectura a la máquina *host* o anfitriona.

3. Descarga e instalación

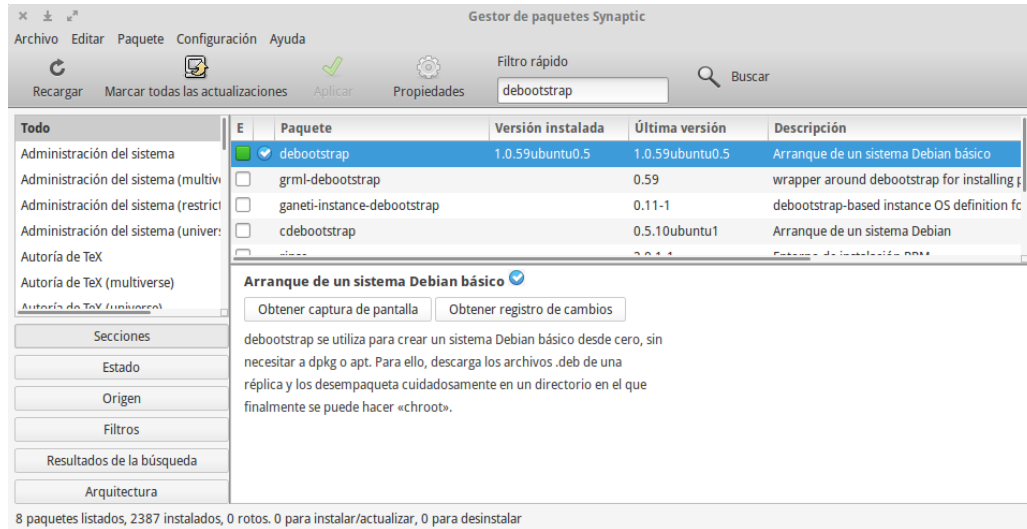


Figura 2: Debootstrap dentro de los repositorios.

Utilizando Ubuntu como sistema host, las herramientas ya mencionadas pueden ser encontradas dentro de los repositorios oficiales del sistema, como se muestra en la figura 2. Pueden ser instalados por consola o por cualquier gestor de paquetes. Se sugiere actualizar los repositorios mediante el comando:

```
$ sudo apt-get update
```

Una vez actualizados, se procede a la instalación de los repositorios:

```
$ sudo apt-get install debootstrap qemu-user-static
```

El proceso durará dependiendo de la velocidad de descarga del internet. Al finalizar, se deberá solicitar permiso de *root* al sistema:

```
$ sudo -s
```

Ahora, se deberá crear el directorio donde se almacenarán los archivos del sistema que se emulará. Se sugiere que el nombre de dicha carpeta haga referencia al tipo de sistema que contendrá. En este caso, se usará el nombre *ARM-Root*.

```
$ mkdir ~/ARM-Root
```

Al finalizar, se procederá a la instalación del sistema:

```
$ debootstrap --foreign --arch=armhf stable ARM-Root
$ cp /usr/bin/qemu-arm-static ARM-Root/usr/bin
$ chroot ARM-Root /debootstrap/debootstrap --second-stage
```

El proceso tardará nuevamente dependiendo de la velocidad del internet. Se sugiere estar pendiente del proceso de descarga e instalación por cualquier error que surgiera durante el mismo¹.

Al finalizar, salga del modo *root* con :

```
$ exit
```

Ya dispone de un sistema Debian ARM totalmente funcional, pudiendo acceder mediante el comando:

```
$ sudo chroot ~/ARM-Root /bin/bash
```

3.1. Configuración del sistema

Se abre una terminal. En ella, ingrese los comandos para acceder al sistema ARM:

```
$ sudo chroot ~/ARM-Root /bin/bash
```

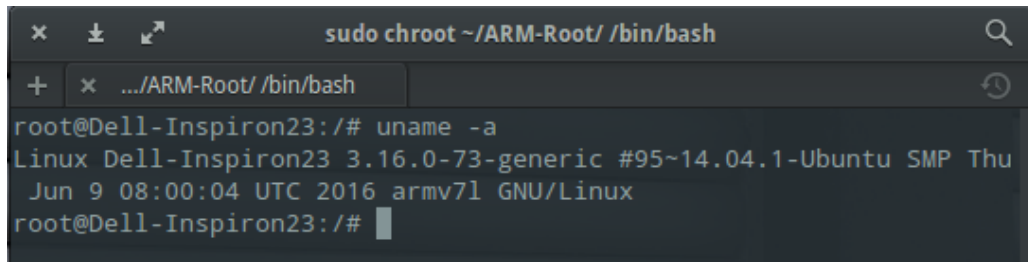


Figura 3: Salida del comando uname.

¹En ocasiones, suelen salir mensajes de que el archivo a descargar no se encuentra ya en el servidor, pero que lo intentará en otro servidor espejo. Es un mensaje normal y no requiere de la atención del usuario.

La terminal se debe mostrar como un usuario root dentro del mismo equipo que usted utiliza. Sin embargo, es un sistema de diferente arquitectura. Escriba el siguiente comando para corroborarlo:

```
$ uname -a
```

El sistema entregará una salida que contiene el nombre del equipo, versión del kernel, entre otros. Lo interesante está en la salida del tipo de arquitectura del sistema, que dice *armv7l*, como se muestra en la figura 3, lo que indica que la instalación fue exitosa. En la ejecución de una terminal común, sin haber ejecutado el comando *chroot*, la salida ofrecerá en el apartado de arquitectura, a *x86_64* ó *x86*, dependiendo del sistema anfitrión.

Ahora, conviene actualizar este sistema ARM e instalar los repositorios necesarios que se van a utilizar en el proyecto actual en la tarjeta ARM física original. Para hacer esto, se verifica la versión de Debian instalada mediante el comando:

```
$ cat /etc/debian_version
```

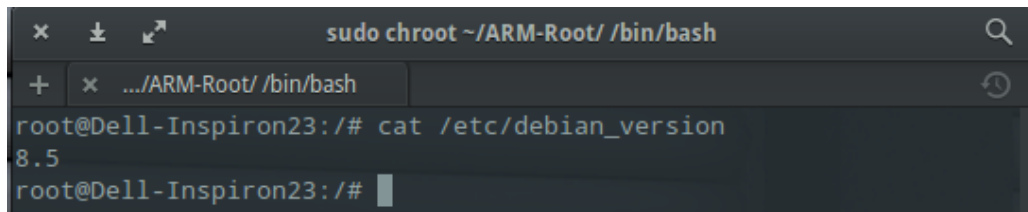
A screenshot of a terminal window with a dark background. The title bar at the top reads 'sudo chroot ~/ARM-Root/ /bin/bash'. Below the title bar, there is a tab labeled '.../ARM-Root/ /bin/bash'. The terminal shows a root prompt 'root@Dell-Inspiron23:/#' followed by the command 'cat /etc/debian_version'. The output of the command is '8.5'. The prompt returns to 'root@Dell-Inspiron23:/#' with a cursor.

Figura 4: Versión del sistema instalado.

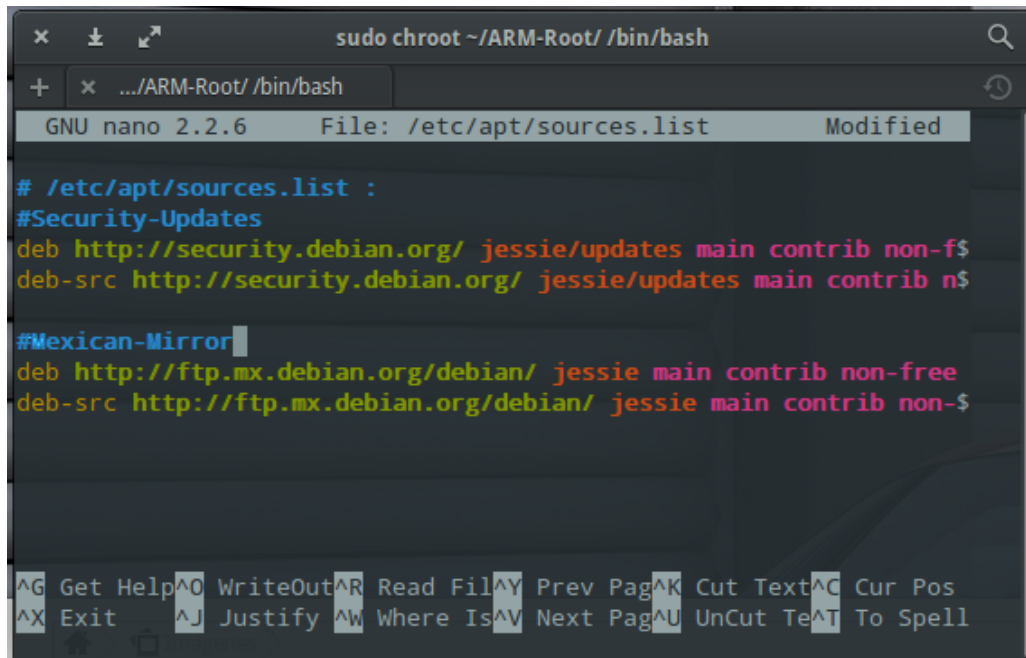
Teniendo el número de versión, se debe conocer con qué nombre fue "bautizada" dicha versión de la distribución. Durante la escritura de este documento, la versión instalada fue Debian *Jessie* (versión 8.*), como se muestra en la figura 4. El archivo */etc/apt/sources.list* es un archivo vacío y en su contenido se debe indicar las direcciones en donde debe buscar por paquetes. Es dependiente de la versión instalada.

En la web LinuxConfig.org² se encuentran listadas las entradas para este archivo en Debian Jessie.

²<https://linuxconfig.org/debian-apt-get-jessie-sources-list>

Para este caso particular, se colocarán las entradas para las actualizaciones de seguridad, y del espejo mexicano. Se hace mediante la entrada del comando:

```
$ nano /etc/apt/sources.list
```



```
GNU nano 2.2.6 File: /etc/apt/sources.list Modified

# /etc/apt/sources.list :
#Security-Updates
deb http://security.debian.org/ jessie/updates main contrib non-free
deb-src http://security.debian.org/ jessie/updates main contrib non-free

#Mexican-Mirror
deb http://ftp.mx.debian.org/debian/ jessie main contrib non-free
deb-src http://ftp.mx.debian.org/debian/ jessie main contrib non-free

^G Get Help ^O WriteOut ^R Read File ^Y Prev Pag ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Pag ^U UnCut Te ^T To Spell
```

Figura 5: Contenido del archivo */etc/apt/sources.list*.

Se vuelve a enfatizar que el contenido de este archivo es totalmente dependiente de la versión de Debian instalada, ya que colocar las fuentes para un sistema diferente puede comprometer el sistema de paquetes de Debian. Una vez colocadas las entradas como muestra la figura 5, se procede a actualizar la lista de repositorios mediante el comando:

```
$ apt-get update
```

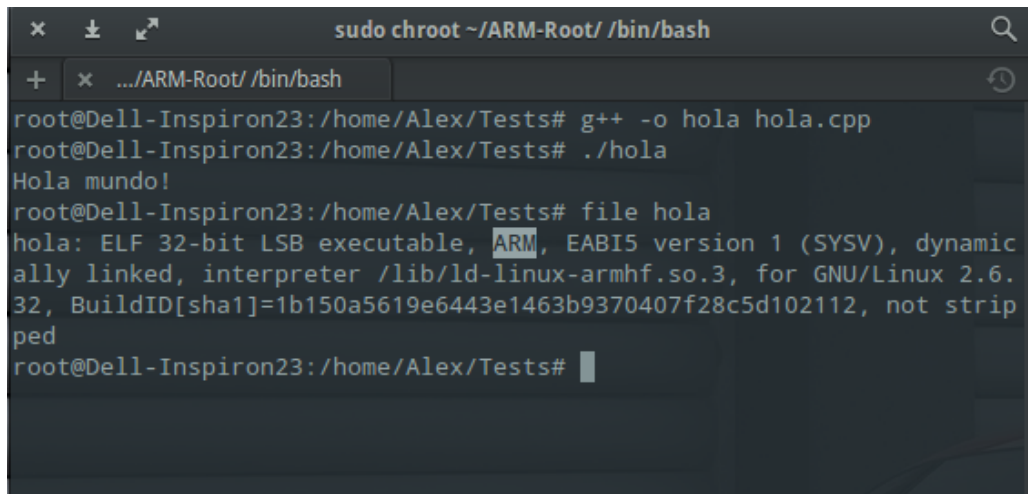
El proceso puede tardar dependiendo de la cantidad de cosas a descargar y de la velocidad a internet. Una vez terminado, se procede a la instalación de los paquetes necesarios para el proyecto en el que se tenga que utilizar este sistema.

3.2. Ejemplo: compilando un programa en C++

Para instalar los compiladores de c y c++, se usa el comando:

```
$ apt-get install build-essential
```

El sistema mostrará la cantidad de bytes a descargar y pregunta si se desea proseguir. Con la tecla y/Y se le indica que continúe y mostrará la descarga. Al acabar, el sistema ya será capaz de compilar programas en los lenguajes ya mencionados.



```
sudo chroot ~/ARM-Root/ /bin/bash
+ x .../ARM-Root/ /bin/bash
root@Dell-Inspiron23:/home/Alex/Tests# g++ -o hola hola.cpp
root@Dell-Inspiron23:/home/Alex/Tests# ./hola
Hola mundo!
root@Dell-Inspiron23:/home/Alex/Tests# file hola
hola: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamic
ally linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.
32, BuildID[sha1]=1b150a5619e6443e1463b9370407f28c5d102112, not strip
ped
root@Dell-Inspiron23:/home/Alex/Tests#
```

Figura 6: Compilación, ejecución y verificación de la arquitectura en que fue compilado un programa de c++.

A modo de prueba, se codificó un archivo en c++ que imprima *Hola mundo*, llamado *hola.cpp* y se procedió a compilarlo y ejecutarlo:

```
$ g++ -o hola hola.cpp
$ ./hola
```

Mediante el comando *file*³, se procedió a verificar que el archivo estaba codificado para arquitectura ARM, como indica la figura 6. Esto significa que el ejecutable *hola* puede ser ejecutado perfectamente en una tarjeta BeagleBone, Raspberry o similares, de arquitectura ARM con sistema GNU/Linux. Como nota, este archivo no podrá ser ejecutado por una terminal de la computadora host sin haber ejecutado el comando *chroot* mencionado al principio de la subsección 3.1, en la página 5, debido a que su arquitectura no soporta la codificación de dicho archivo.

³ Instalable mediante: \$ apt-get install file