En proceso de traducción.rramp

Soporte Qemu en Slackware ARM

Este documento describe el proceso de instalación de **Slackware ARM** dentro de QEMU.

- QEMU es un emulador de sistema completo que puede emular una amplia gama de arquitecturas de hardware reales. En este caso usaremos QEMU para emular a ARM Ltd. Tarjeta de desarrollo “Versatile Express”.
- QEMU proporciona una plataforma que permite a un sistema operativo creer que se está ejecutando en hardware real.
- QEMU se ejecuta sobre tu pc o servidor Slackware. En muchos casos esto puede ser una máquina de escritorio. Aunque, es posible ejecutar QEMU en un servidor sin monitor y usar VNC para gráficos.

¿A quién va dirigido Slackware ARM en QEMU?

El objetivo de instalar Slackware ARM dentro de QEMU es permitir a las personas que no tienen hardware ARM probar Slackware ARM.

Mientras que QEMU es un emulador excelente, es muy lento comparado con el hardware real de ARM. QEMU se utilizó durante aproximadamente un año para desarrollar la mayor parte de Slackware ARM versión 12.2. Debido a la lenta velocidad de QEMU, distcc puede ser usado en varias máquinas x86 para acelerar las compilaciones. Es posible utilizar Slackware ARM de forma efectiva en QEMU, sin embargo, tenga en cuenta que no presentará la mejor experiencia de usuario debido a su falta de velocidad. Lo ideal es que Slackware ARM se ejecute en hardware ARM real.

Este documento es un trabajo en curso y su objetivo es la publicación de ARM, Slackwarearm-current.

Documentación para Slackware ARM 14.0, 14.1, y 14.2, puede ser encontrado aquí:

- Slackwarearm-14.0
- Slackwarearm-14.1
- Slackwarearm-14.2

Suposiciones del entorno de instalación

Se hacen varias suposiciones para escribir este documento.

- La máquina anfitriona está ejecutando una instalación completa de Slackware x86 o x86_64.
- Usted tiene acceso de root en la máquina anfitriona y puede compilar software.
- La máquina anfitriona Slackware y el emulador QEMU están sobre una red segura local (LAN).
- Puede exportar un directorio compartido (NFS) en la máquina anfitriona que comparte el árbol de Slackware ARM con el huésped de QEMU.
- El directorio raíz del directorio exportado mediante NFS en la máquina anfitriona se encuentra en /export.

Estas directrices se pueden cumplir con una máquina anfitriona Slackware x86/x86_64. Esta única máquina anfitriona puede servir el contenido de NFS, así como alojar la instalación de Slackware ARM dentro de QEMU.

Prerequisitos para una máquina anfitriona Slackware x86

1. Descarga el árbol de paquetes de distribución **Slackwarearm** con rsync.
2. Complete una ubicación con los archivos QEMU de Slackware ARM.
3. Configurar el Sistema de Archivos de Red (NFS) en el Host x86 de Slackware.
4. Instalar QEMU y el software del compilador de árbol de dispositivos en el host x86 de Slackware.
5. Configurar los permisos de QEMU en el host x86 de Slackware.
6. Crear una imagen del disco QEMU usando los scripts de ayuda.

Descargar Slackware ARM

Descargar Slackware ARM a tu anfitrión Slackware x86. En este tutorial es utilizado rsync para un espejo(mirror) Slackware ARM. Antes de ejecutar rsync, haga que su usuario haya leído, escrito y ejecutado permisos dentro del directorio /export.

```
mkdir -p /export/slackwarearm
cd /export/slackwarearm
rsync --exclude '*/source/*' --delete -Pavv
ftp.arm.slackware.com::slackwarearm/$SLACKREL .
```

Si tu quieres puede usar una navegador web, wget, o lftp para descargar Slackware ARM, el espejo (mirror) esta aquí.

Las variables $SLACKREL se refieren a el árbol Slackware ARM que descargaste. Si elegiste Slackwarearm-14.2, la url para rsync sera ftp.arm.slackware.com::slackwarearm/slackwarearm-14.2

Complete los archivos y directorios de Slackware ARM

Para arrancar el instalador de Slackware ARM necesita crear un directorio que almacene el Kernel y el disco RAM inicial. También necesita algunos scripts de ayuda para ejecutar QEMU. En este tutorial, todos estos archivos se almacenarán en /export/armhost.

Copiar el kernel y el disco RAM:

```
mkdir -p /export/armhost
cp -fa /export/slackwarearm/$SLACKREL/kernels/armv7/{zImage*,initrd*,dtb}
/export/armhost/
cp -fa /export/slackwarearm/$SLACKREL/isolinux/initrd-armv7.img
/export/armhost/
```

Descargar y copiar los scripts de ayuda de QEMU:

```
# cd /tmp ; mkdir qemu ; cd qemu
# rsync -Prlvv --delete
ftp.arm.slackware.com::slackwarearm/boardsupport/qemu/$SLACKREL .
# cd $SLACKREL
# cp -fav helper-scripts/* /export/armhost/
```

Usted puede buscar los scripts de ayuda aquí si no desea utilizar rsync para descargarlas.

La variable $SLACKREL se refiere al árbol Slackware ARM tree que descargaste. Si elegiste Slackwarearm-14.2, el camino (path) puede ser /export/slackwarearm/slackwarearm-14.2. La URL para rsync para los scripts de ayuda de QEMU helper scripts son:

```
ftp.arm.slackware.com::slackwarearm/boardsupport/qemu/slackwarearm-14.2
```

Configuración de los servicios de sistema de archivos(NFS) de red

El anfitrión x86 de Slackware necesita ser configurado apropiadamente para poder ejecutar el proceso de instalación de Slackware arm dentro de QEMU. La forma más fácil y eficiente de hacerlo es configurando un recurso compartido de Sistema de archivos de red (NFS) en el equipo host. Este recurso compartido NFS será utilizado por el cliente Slackwarearm (que se ejecuta dentro de QEMU) para acceder al árbol de paquetes de distribución de Slackware en el host. Los servicios de NFS no son la única manera de servir el árbol de paquetes de distribución de Slackware al cliente de QEMU. Otros métodos son un poco más lentos cuando se usan con redes emuladas de QEMU. Como resultado, el servicio NFS se utiliza para este tutorial.

Necesita conocer la configuración de red de QEMU para acceder al recurso compartido NFS. Se recomienda que utilice el modo NAT de conexión en red con QEMU. El modo NAT permite el acceso directo al recurso compartido NFS del host x86 de Slackware a través de una red virtual alojada en QEMU. Más adelante en esta guía se discute más información sobre la red virtual y el direccionamiento asignado.

Con un editor de texto, como root, agregue los siguientes host's al archivo /etc/exports:

```
# QEMU guest virtual IP address
/export/slackwarearm
10.0.2.15(insecure,ro,nohide,root_squash,sync,no_subtree_check)
```

El modo NAT no permite el acceso directo a la red de área local física. Tendrá que configurar una interfaz de red puenteada si decide que necesita acceso directo a la red. El modo NAT será suficiente para la mayoría de los usuarios.

Utilice algo similar a lo siguiente si planea utilizar una interfaz de red puenteada:

```
/export/slackwarearm
xxx.xxx.xxx.x/255.255.255.0(insecure,ro,nohide,root_squash,sync,no_subtree_c
heck)
```

Reemplace xxx.xxx.x.x/255.255.255.0 y equipáralo con la configuración de su red. (Ejemplo: 192.168.1.0/255.255.255.0)

Active el recurso compartido NFS ejecutando lo siguiente como root:

```
# chmod +x /etc/rc.d/{rc.rpc,rc.nfsd}
# /etc/rc.d/rc.nfsd start
# exportfs -va
```

Instalar QEMU y el compilador de árbol de dispositivos

La instalación de SlackBuildes no está fuera del alcance de este artículo. Si necesita ayuda con la instalación de QEMU o del compilador de árboles de dispositivos, por favor, consulte la sección Paginas HOWTO de SlackBuilds.org. Dicho esto, hay algunas recomendaciones:

- Si estas corriendo Slackware-current puedes instalar QEMU y omita la instalación del paquete dispositivo-árbol-compilador. Slackware-current ya incluye el paquete de compilador de árbol de dispositivos en una instalación completa.
- Asegúrese de haber realizado una instalación **completa** de Slackware en su host x86 antes de instalar estos SlackBuilds.
- Los usuarios de Slackware 14.0, 14.1 y 14.2 necesitan instalar paquete de compiladores de dispositivos de árbol de SlackBuilds.org antes de instalar QEMU.
- Puede descargar e instalar QEMU de SlackBuilds.org.
- No tendrá la capacidad en los sistemas 14.0, 14.1 y 14.2 de emular la arquitectura ARM en QEMU si no instala primero el paquete del compilador del árbol de dispositivos.

Permisos para QEMU

Hay algunos permisos que deben ser establecidos una vez que haya instalado correctamente QEMU en su sistema. El cliente QEMU se iniciará ejecutando el binario /usr/bin/qemu-system-arm. Este binario necesita permisos de root para ser ejecutado. Puede ejecutar este binario con sudo editando /etc/sudoers. Usar sudo es la opción más segura si tiene varios usuarios en su sistema. Si usted es el único usuario de su sistema, es suficiente con configurar el permiso setuid como root. Ajustar estos permisos permitirá a un usuario normal configurar y arrancar el huésped de QEMU sin tener que iniciar sesión como root.

Para establecer el permiso setuid root es necesario que inicie sesión como root. Como root ejecute los siguientes comandos:

```
# chmod +s /usr/bin/qemu-system-arm
```

Si planeas configurar QEMU para que use una red puente, vas a necesitar configurar los permisos setuid para /sbin/ifconfig y /sbin/brctl.

```
# chmod +s /sbin/{ifconfig,brctl}
```

Crear una imagen de disco para QEMU

Antes de arrancar el instalador de Slackware ARM en QEMU, debe crear una imagen de disco que actúe como una emulación de una SD Card. Este disco de imagen es usado para emular el controlador MMC en Slackware ARM. Anteriormente copió los scripts de ayuda de Slackware ARM en /export/armhost/. Dentro de este directorio hay un script, **makeimg** . Este script crea una imagen de disco de 15GB automáticamente en /export/armhost/ cuando se ejecuta. Inicialmente, todo lo que necesita hacer es ejecutar este script. Cambia al directorio donde copiaste los scripts de ayuda y ejecuta **makeimg**:

```
cd /export/armhost
./makeimg
```

Para referencia, este es el script **makeimg**:

```
# Create the QEMU disk image - the emulated SD card.

IMG=sdcard.img
SIZE=15G

rm -f $IMG
qemu-img \
   create \
   $IMG $SIZE
```

Tenga en cuenta que una vez que haya instalado Slackware ARM en esta imagen de disco debe moverlo a un directorio diferente para su almacenamiento, o se arriesga a que se destruya cuando **makeimg** se ejecute en un momento posterior.

Configuración de red para QEMU

Esta sección cubre el proceso de configuración de redes de invitados para QEMU. Dos tipos de procesos son descriptos. *Network Address Translation mode (NAT)* es la primera y la forma recomendada para obtener una red red funcional en los huéspedes de QEMU. La segunda forma es el *modo puente*. El modo NAT no permite el acceso directo a la red física del host x86 Slackware y el modo puente sí. Es mejor utilizar el modo puente si planea realizar operaciones de red más avanzadas que requieran acceso completo al host y a la red física del host. **La mayoría de los usuarios querrán usar el modo NAT**.

Existen muchas formas diferentes para configurar la red de invitados para QEMU. Este documento solo cubrirá las funcionalidades de QEMU para que Slackware ARM arranque. Refiérase a las paginas del manual o a la documentación de QEMU si necesitas una explicación más detallada.

Later on when you boot the Slackware ARM installer in QEMU you may need to modify the **txqueuelen** for your network interfaces. This is because large Slackware packages time out while being downloaded from the NFS share on the host. This happens because QEMU emulation is very slow. The NFS daemon on your host machine occasionally shuts down the network socket before large packages (rust, kernel-firmware, etc) finish being copied to the SD Card. Setting the txqueuelen to **10000** for all network interfaces should be sufficient to prevent this anomaly. The following command seems to resolve this issue:

```
ip link set eth0 txqueuelen 10000
```

Run this command for each network interface actively used by QEMU.

Modo de red NAT para QEMU

El modo NAT no requiere una configuración adicional sobre una máquina Slackware x86 o en la máquina invitada QEMU. Aquí hay un ejemplo abreviado de un invitado de QEMU que está siendo lanzado con el modo de red NAT:

```
# cd /export/armhost
# qemu-system-arm -nographic \
   -m 1024 \
   -M vexpress-a9 \
   -k en-us \
   -net nic \
   -net user
   ..snip..
```

The **-net nic** and **-net user** options enable QEMU to start the Slackware ARM guest with NAT mode networking enabled. These settings are documented further in both the **installer_launch** and **disk_launch** helper scripts.

With NAT mode enabled, QEMU launches a virtual network of 10.0.2.0/24. The QEMU guest will be assigned the IP address 10.0.2.15. The guest can access the Slackware x86 host at 10.0.2.2 and the QEMU DNS server runs at 10.0.2.3. QEMU does not have direct access to the host's Local Area Network. This means that the QEMU guest isn't assigned a physical IP address by your router DHCP service. You will not be able to ping the QEMU guest from the host machine but the guest should be able to ping the host machine at 10.0.2.2. The QEMU guest should be able to access the internet and communicate with the host machine.

Modo de red puente para QEMU

The best way to set up a bridged network interface for QEMU is with the provided helper script, rc.local-additions. If you choose to take this route, you need to disable the NetworkManager service. NetworkManager comes with Slackware, but it is not developed by Slackware. As a result, the process of configuring a bridge with NetworkManager is not supported in this guide.

You can find the helper script with comments online here: rc.local-additions

The following commands must be executed as root to disable NetworkManager:

```
# /etc/rc.d/rc.networkmanager stop
# chmod -x /etc/rc.d/rc.networkmanager
```

The following network settings are assumed for the **Slackware x86 host machine**. Adjust these values throughout the remainder of this guide if you use different network settings.

```
Default Gateway: 192.168.1.1
Static IP address: 192.168.1.2
Network Mask: 255.255.255.0 / 192.168.1.0/24
Name server: 192.168.1.1
```

You need to edit the *qemu-network-tun.sh* helper script for QEMU. It is required to bring up the QEMU guest network interface. It should exist in /export/armhost. Change the IP address listed in $BRIDGEIP to match your network settings.

File: /export/armhost/qemu-network-tun.sh

```
#!/bin/sh

# This is the IP of 'tap0' on the Slackware/x86 host:
BRIDGEIP=192.168.1.4

modprobe tun
/sbin/ifconfig $1 $BRIDGEIP netmask 255.255.255.0
/sbin/brctl addif br0 $1
```

Here is the relevant portion of the rc.local-addtions script that requires modification. This script is meant to replace on the host machine /etc/rc.d/rc.local. Changes depend on your network settings:

```
.. snip ..

# Turn on the bridge.  Note that this is a different IP from
# the one specified in your qemu-network-bridge.sh script
# in your 'armhost' directory on your Slackware x86 box.
# You need a bridge IP, a tunnel (tap0) IP, and then another
# IP which is assigned to the Slackware ARM host (by Linux inside QEMU)
# to its own eth0.
# I tried bringing this up after eth0 but the bridge didn't work.
# I don't know why that is!
ifconfig br0 192.168.1.3 up

# Put back the original IP for eth0:
ifconfig eth0 192.168.1.2 up

#
route del default
route add default gw 192.168.1.1
```

These network settings assume static IP addressing. The br0 interface is the bridged network interface. The eth0 interface is the host machine network interface that allows the host machine to retain network connectivity. The default gateway, 192.168.1.1, typically points to the network gateway on the physical network.

Copy rc.local-addtions to /etc/rc.d/rc.local once you finish editing it. Then mark it executable.

```
# cp /path/to/rc.local-additions /etc/rc.d/rc.local
```

```
# chmod +x /etc/rc.d/rc.local
```

At this point it is recommended reboot your Slackware x86 host to assure the settings in rc.local are in use and that NetworkManager is completely disabled.

You may need to edit the /etc/resolv.conf file on the host. Add in the IP addresses of your preferred primary and secondary name server(s) since you are not receiving these IP addresses by other means. The name server is 192.168.1.1 (or the default gateway) in this tutorial

Modify the launcher helper scripts once you are certain your host has the appropriate network settings. Pay close attention to the **$NETTYPE** variable. Details about how to use this variable are documented in both the **installer_launch** and **disk_launch** helper scripts. Edit the **$MACADDR** variable for each QEMU instance if you are running more than one Slackware ARM guest at once.

Install Slackware ARM

I will assume that you are now in X Windows, running as your normal user account, and that you followed the steps outlined earlier in this document. As stated earlier, QEMU runs extremely slow when emulating the ARM architecture. Depending on your hardware set up it may take several hours or more for the Slackware installer to copy all packages to the emulated disk. If the Slackware installer appears to be unresponsive, check your system process monitor (top or htop) to see if the QEMU process is still active. A good sign that QEMU is still active is that a single CPU core is operating at 100 percent.

The Slackware ARM installer is mostly identical to the Slackware x86 installer. There is no learning curve to install Slackware ARM if you have installed Slackware before.

Booting the Installer

In order to boot the installer you will need to configure and execute the **installer_launch** script within a terminal window.

```
cd /export/armhost
./installer_launch
```

The **installer_launch** script can be found here.

You will see some warnings from QEMU about being unable to open audio and video devices. Those warnings can safely be ignored. Next you will see the Linux kernel boot messages and eventually the installer asking about what key map you want to use. Once you select your key map and log in to the system you will notice that the installer obtains an IP address via DHCP. The DHCP assigned IP address and resulting network configuration relies heavily on how you set up your networking on the host machine. If QEMU does not assign an IP address to the guest, then you need to go back and verify your network settings are configured appropriately.

Partitioning

The emulated SD Card created with the **makeimg** command is a blank image. You will have to partition this SD Card with the installer. It is best to keep the partition scheme simple in our case. It is recommended that you create a 200MB swap partition and to allocate the rest of the disk to the root partition. You can use the **fdisk** or **cfdisk** tools to create the partitions.

Example partitioning scheme:

```
/dev/mmcblk0p1 - 200MB swap
/dev/mmcblk0p2 - the rest of the disc, "Linux" - type 83.
```

Setup and Configuration

Run the **setup** command at the shell prompt after you exit the partitioning tool. Make the installer aware of your swap parition and root partition. It is recommended that you select the ext4 file system when you format the root partition. Next you will be prompted to select the source media. Choose option 4, **Install from NFS (Network Filesystem)**. Enter the IP address of your Slackware x86 host. Enter the path to the NFS mounted share.

The full path of the NFS share is required:

```
Enter the IP address: 192.168.1.2 # Host machine IP address
Enter the directory: /export/slackwarearm/$SLACKREL/slackware
```

Following that, you will be prompted for package selection. Slackware ARM has all of the standard Slackware packages apart from those which are x86 only. It is highly recommended that you do a **full** installation to satisfy all system dependencies. Please be patient, this is the most time consuming part of the installation process.

After installation has finished, running 'MKFONTDIR AND MKFONTSCALE UPDATE' takes a long time.

At the Network Setup screen it is best to select the DHCP option. The DHCP option best compliments QEMU's NAT mode and both bridged mode networking options. The only reason not to select DHCP is if your physical network uses static IP addressing.

Next you will reach the Window Manager selection for the X Windows server. It is recommended that you select a light weight window manager, such as Fluxbox or WindowMaker. KDE and Xfce are not very useful within the QEMU guest due to speed constraints.

Post-Installation

Once you complete the installation process you should drop to a shell prompt to configure the SSH Daemon. By default, OpenSSH does not allow root to log in with a password. This is a security concern. You may want to think about this carefully if your device is connected directly to an untrusted network. It is best to make a user account for remote connections to the SSH service and escalate privileges locally with “su” or “sudo”. If you wish to use root to log in remotely, follow these steps:

1. Opt to drop in to a 'shell' when you exit from the installer
2. At the shell, enter:

```
# sed -i 's?^#PermitRootLogin.*?PermitRootLogin yes?g'
/mnt/etc/ssh/sshd_config
# poweroff
```

This completes the installation process of Slackware ARM within QEMU.

Boot Slackware ARM with QEMU

Congratulations for making it this far! The next step is booting into your fresh installation of Slackware ARM. Locate the **disk_launch** helper script in /export/armhost and modify it to fit your needs.

```
# cd /export/armhost
# vi disk_launch
```

This script has a few variables you may want to change.

- **ROOTFSTYPE** - root file system type, ext4 is the default
- **ROOTFSDEV** - location of the root partition within the SD Card image
- **KEYBOARD** - keyboard locale you wish to use, typically the same as what you chose during installation
- **NETTYPE** - network configuration, NAT mode or bridged mode

The **disk_launch** script for Slackwarearm-current can be found online, here.

The first boot will take quite a while. This is due to the fact that Slackware will generate the font cache for the first time. Start up QEMU by executing the disk_launch script.

```
# ./disk_launch
```

Assuming all is well, you can begin using Slackware ARM just as you would any other Slackware installation.

Slackware ARM Interfaz gráfica de usuario

Trabajo en progreso

Esta sección discutirá los aspectos positivos y negativos de ejecutar X Windows, qué administrador de ventanas o entorno de escritorio usar, y las formas en que puede iniciarlo.

Fuentes

- Escrito originalmente por: Stuart Winter.
- Fuente original: http://ftp.arm.slackware.com/slackwarearm/boardsupport/qemu/.
- Traducido por: — *rramp 2019/07/31 03:10 (UTC)*.

- Modificado y mantenido por: Brenton Earl (mralk3).

howtos, hardware, arm, user mralk3