Read-only diskless filesystem with Debian 7.0 CTRL.01.01.001

Sébastien Blanchet

Oct 13, 2014

Contents

Chapter 1

Build NFS root

1.1 Create NFS root

1.1.1 Create the host computer

A Debian host computer is required to create the nfsroot filesystem. In theory, you can use any Debian computer, whatever the release or the architecture are. But in practice it is more convenient to use the same release and archictecture for both. So create a new virtual machine with VirtualBox and install Debian-7.0-amd64 into.

1.1.2 Create a bootstrap

`debootstrap` creates a basic Debian system.

```
# aptitude -y install debootstrap
# mkdir /home/nfsroot
# debootstrap wheezy /home/nfsroot http://ftp2.fr.debian.org/debian
```

The mirror URL is optional, but it downloads faster if you select a mirror near you. See manual `debootstrap(8)` for details.
Now we customize this basic Debian system in `/home/nfsroot`.

```
# export NFSROOT=/home/nfsroot
```

1.1.3 /etc/fstab

Listing 1.1: $NFSROOT/etc/fstab

```
proc                /proc      proc    defaults   0 0
/dev/nfs            /          nfs     tcp,nolock 0 0
none                /tmp       tmpfs   defaults   0 0
```

```
none                  /var/tmp   tmpfs   defaults   0 0
none                  /media     tmpfs   defaults   0 0
none                  /var/log   tmpfs   defaults   0 0
nfssrv:/path/to/home  /home      nfs     tcp,nolock 1 2
```

The `nolock` nfs option is very important, otherwise sqlite does not work on these mount points. See `man 5 nfs`, for other nfs options.

1.1.4 Directory /etc/default

See manuals `rcS(5)` and `tmpfs(5)` for details.

```
# echo ASYNCMOUNTNFS=no >> ${NFSROOT}/etc/default/rcS
# echo RAMTMP=yes >> ${NFSROOT}/etc/default/tmpfs
```

1.1.5 /etc/apt/sources.list

Point to the nearest mirror, and add `contrib` and `non-free` repositories.

Listing 1.2: /etc/apt/sources.list

```
deb http://ftp2.fr.debian.org/debian/ wheezy main contrib non-free
deb-src http://ftp2.fr.debian.org/debian/ wheezy main  contrib non-free

deb http://security.debian.org/ wheezy/updates main contrib non-free
deb-src http://security.debian.org/ wheezy/updates main contrib non-free
```

Host and nfsroot must share the same configuration:

```
# cp /etc/apt/sources.list ${NFSROOT}/etc/apt/sources.list
```

1.1.6 /etc/hosts

Remove all IPv6 lines and add IP for each NFS client.

Listing 1.3: $NFSROOT/etc/hosts

```
127.0.0.1       localhost
192.168.0.186   pc-client1
192.168.0.187   pc-client2
192.168.0.188   pc-client3
```

1.1.7 /bin/whereami

Create a script `/bin/whereami` to setup client hostname from `/etc/hosts`

Listing 1.4: $NFSROOT/bin/whereami

```
#!/bin/bash
#finds node's hostname based on matching ip in /etc/hosts

grep `ifconfig                                  \
    | grep 'inet addr:'                         \
    | grep -v '127.0.0.1'                       \
    | /usr/bin/cut -d: -f2                      \
    | /usr/bin/awk '{ print $1}'` /etc/hosts \
    | /usr/bin/awk '{print $2}'
```

Add executable bit:

```
# chmod +x ${NFSROOT}/bin/whereami
```

Modify `$NFSROOT/etc/init.d/hostname.sh` to replace

```
    [ -f /etc/hostname ] && HOSTNAME="$(cat /etc/hostname)"
```

by

```
    [ -f /etc/hostname ] && HOSTNAME="$(/bin/whereami)"
```

1.1.8 Configure /etc/mtab

```
# ln -s /proc/mounts ${NFSROOT}/etc/mtab
```

1.1.9 Configure reboot

Tweak reboot scripts to avoid cutting down the network interface.

```
# sed -i.orig  -e 's/reboot -d .*/reboot -d -f/g' \
    ${NFSROOT}/etc/init.d/reboot
# sed -i.orig  -e 's/halt -d.*/reboot -d -f/g' \
    ${NFSROOT}/etc/init.d/halt
```

1.1.10 Configure the root user

```
# chroot /home/introot
# passwd
```

1.1.11 Install additionnal firmware

It must be done before rebuilding initrd.

```
# aptitude update
# aptitude install firmware-linux
```

1.1.12 Install additionnal driver

It must be done before rebuilding initrd.
In this example we add an updated driver to support Realtek 8111G.
Download `r8168-8.037.00.tar.bz2` from `http://www.realtek.com`

```
# tar xf r8168-8.037.00.tar.bz2
# cd r8168-8.037.00
# bash ./autorun.sh
```

The `autorun` script does:

- build the new driver
- disable the linux driver `r8169`
- install the new driver `r8168` in `/lib/modules/‘uname -r‘` so it will be automatically present when we rebuild the initrd.

1.1.13 Build a PXE initrd

The default initrd image is not suitable for diskless computer, so rebuild a new one.

```
# mkdir -p  /var/lib/tftpboot/
# cp -r /etc/initramfs-tools /etc/initramfs-pxe
# echo BOOT=nfs >> /etc/initramfs-pxe/initramfs.conf
# mkinitramfs -d /etc/initramfs-pxe \
    -o /var/lib/tftpboot/initrd.pxe-‘uname -r‘ ‘uname -r‘
```

Initrd customization

You can uncompress the initrd image to edit its content:

```
# mkdir initrd
# cd initrd
# gunzip2 -dc initrd.pxe-‘uname -r‘ | cpio -i
# #MODIFY the files here
# find . | cpio -c -o | gzip -9 > initrd.pxe-‘uname -r‘
```

1.1.14 Driver blacklisting

During a Linux diskless booting process, the NFS root filesystem **must** be mounted through the PXE booting interface **only**. But during the initrd step, the kernel does not know which interface has been used for PXE booting, so it will use the first detected interface instead.

For example if you add a Broadcom network card into a Beckhoff computer, you may encouter this kind of problem:

- the computer can boot only from its internal Intel network interfaces.
- `initrd` detects the Broadcom before the Intel network interfaces, so it tries to mount the NFS root filesystem through the wrong interface. So it fails.

The solution is to blacklist the broadcom driver `tg3` during the init. There are two ways to do it:

- by adding `blacklist=tg3` to the kernel command line in the pxeconfig file. It is the easier and more flexible solution.
- by editing the `blacklist.conf` file inside the initrd

Editing blacklist.conf inside initrd file

```
# mkdir initrd
# cd initrd
# bunzip2 -dc initrd.pxe-‘uname -r‘ | cpio -i
# echo "blacklist tg3" >> etc/modprobe.d/blacklist.conf
# find . | cpio -c -o | bzip2 -9 > initrd.pxe-‘uname -r‘
```

Note: With the Debian 7.0 i386 3.2.0-4-rt-686-pae kernel, this procedure does not work. The kernel is unable to mount the new initrd (it works only with the initrd created with mkinitramfs).

1.2 Prepare NFS and TFTP server

- Transfer nfsroot to the nfs server.
- Transfer kernel and initrd to the tftp server.

Create a pxeconfig file

Listing 1.5: $NFSROOT/etc/hosts

```
# boot diskless computer with debian wheezy
default menu.c32
prompt 0
menu title pc-client

ontimeout linux-3.2.0-3-amd64-wheezy
timeout 50

label linux-3.2.0-3-amd64-wheezy
  menu label linux-3.2.0-3-amd64 wheezy
  kernel vmlinuz-3.2.0-3-amd64
```

```
append root=/dev/nfs initrd=initrd.pxe-3.2.0-3-amd64    \
       nfsroot=nfsserver:/path/to/nfsroot  ro panic=60
ipappend 1
```

Note: the backslash in the `append` line is only for printing purpose, to show that the line continue. But in reality, all the arguments *must* be on the same line.