Root on NFS: Running Linux on a diskless computer

Eli Billauer

January 31st, 2011

This work is released under Creative Common’s CC0 license version 1.0 or later. To the extent possible under law, the author has waived all copyright and related or neighboring rights to this work.
1. Introduction

2. Setting up the filesystem

3. Handling the boot process

4. Wrapping up
Introduction
The goal

- A bare motherboard with no storage attached
- System boots from network
- The root directory (and hence all files) are mounted via network
- Keep it simple
- A full Fedora 12 computer supplies the infrastructure
- The diskless runs CentOS 5.5, mainly for ssh access
- No initramfs in the boot process
Now why you wanna go and do that?

- It’s cool
- It’s not very expensive
- Hardware development along with writing a kernel driver
- Very easy to access diskless computer’s files
- Chances motherboard will be damaged
- System crashes are likely. No fsck, please.
- Virtualization doesn’t solve either
Low-end shopping list

- CPU: AMD Sempron 140 2.7 GHz – 150 NIS
- Gigabyte DDR2 1066 MHz motherboard – 230 NIS
- Kingston 1GB DDR2 800 MHz RAM – 100 NIS
- CPU Fan (Arctic Alpine 64 Pro) – 60 NIS
- Golden Tiger 400W Power supply – 105 NIS
- Total: 645 NIS
- I would invest more in the power supply (at least)
- No box necessary. It’s just two parts when assembled.
### My own shopping list

- **CPU:** Dual Core Intel CPU E5700 3.0 GHz – 270 NIS
- **Gigabyte DDR2 1333MHz motherboard** – 250 NIS
- **Kingston 1GB DDR2 800 MHz RAM** – 100 NIS
- **CPU Fan (Alpine 11)** – 45 NIS
- **Thermaltake 450W Power Supply** – 250 NIS
- **Total:** 915 NIS in the high-end hardware configuration
The naked motherboard gotchas

- Consider letting the shop assemble the components
- ATX power supply alone = dead
- ATX power supply connected to motherboard = dead
- Fake a power button
- Enable PXE and immediate power-up in BIOS
- On-board Gigabit ethernet adapter doesn’t talk well with Edimax switch (enable NIC’s BOOT ROM in BIOS?)
- Gigabyte motherboard won’t boot from external NIC
- Only recent kernels support recent on-board NIC
Booting: Let’s see we’re in sync

- Boot loaders
- The kernel image
- The Initial RAM disk ("initrd/initramfs")
- Kernel parameters
- NFS
- PXE (Preboot eXecution Environment)
- DHCP
- TFTP
The boot process

- BIOS making DHCP request on bus
- DHCP server supplies an IP address and initial file to download (/pxelinux.0)
- BIOS loads pxelinux.0 through TFTP. (on the server it’s /var/lib/tftpboot/pxelinux.0) and runs it.
- PXELINUX tries to load several configuration files from TFTP server, ending up with /pxelinux.cfg/default.
- PXELINUX loads /menu.c32, a GRUB-like menu application, which uses the configuration file to set up menus, defaults, etc.
- PXELINUX loads kernel image (and initrd if required) and runs the kernel with kernel parameters.
- Kernel configures NIC with a renewed DHCP request
- Kernel mounts NFS as root read-only
- Boot continues normally (run /sbin/init)
Setting up the filesystem
NFS – The Nightmare File System

- No cache
- No Authentication
- But it’s natively UNIX (permissions etc.)
- No SELinux labels
- Kernel mounts it as root natively (no need for initrd)
- Has configuration tools on Gnome
- But this line in /etc/exports does the job for sharing
  
  `/cent55_root/
  
  /cent55_root 10.1.1.0/255.255.255.0(rw,sync,no_root_squash)
- no_root_squash means anyone can write as root.
- But access is restricted.
Preparing the root filesystem

- Basically, install a Linux system
- My original idea: Run installation over NFS as root
- Was that naive.
- Solution: Install Linux on a virtual machine (QEMU/KVM)
- Then copy all files to the mounted NFS
- Was that slow.
- I could fiddle with paravirtualization
- But on a system I’m just going to throw away?
More yak shaving

- The goal: Copy GB’s of files
- Shut down the virtual machine
- Mount the partition as a loop device
- Copy files
- But how do you mount a partition from a disk image?

```bash
{ tar -c --one-file-system --to-stdout --preserve * ; } |
{ cd /path/to/new/root && tar --preserve -v -x ; }
```
How to mount a partition from a disk image

# fdisk -lu linux.img
You must set cylinders.
You can do this from the extra functions menu.

Disk linux.img: 0 MB, 0 bytes
255 heads, 63 sectors/track, 0 cylinders, total 0 sectors
Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0x00000000

<table>
<thead>
<tr>
<th>Device</th>
<th>Boot</th>
<th>Start</th>
<th>End</th>
<th>Blocks</th>
<th>Id</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>archlinux.img1 *</td>
<td>63</td>
<td>208844</td>
<td>208844</td>
<td>104391</td>
<td>83</td>
<td>Linux</td>
</tr>
<tr>
<td>archlinux.img2</td>
<td>208845</td>
<td>738989</td>
<td>738989</td>
<td>265072+</td>
<td>82</td>
<td>Linux swap / Solaris</td>
</tr>
<tr>
<td>archlinux.img3</td>
<td>738990</td>
<td>6072569</td>
<td>6072569</td>
<td>2666790</td>
<td>83</td>
<td>Linux</td>
</tr>
<tr>
<td>archlinux.img4</td>
<td>6072570</td>
<td>6281414</td>
<td>6281414</td>
<td>104422+</td>
<td>83</td>
<td>Linux</td>
</tr>
</tbody>
</table>

# mount -o loop,offset=$((512*738990)) linux.img mnt/
Handling the boot process
Cobbler

- A package for people who do mass-installations of Linux systems, on bare metal or virtually
- One of the techniques offered is network installation with PXE boot.
- Integrating the management of several applications: DHCP, TFTP, PXE, Kickstart, Koan, DNS, HTTPD web server for holding local repositories
- Not really intended for a single diskless computer
- Offers good advice with “cobbler check”
Cobbler (cont.)

- The cobbler application is a frontend for cobblerd, which must be running
- cobblerd has personality and free will, and doesn’t care if we CTRL-C its frontend
- Written in Python
- Error messages follow Python tradition
- Go “du -sh /var/www/cobbler/” every now and then. Expect surprises.
- Don’t try to garden its managed directories yourself.
- Was good to kick things off, and then put aside
Kicking off

# yum install cobbler
# yum install dhcp

- Edit /etc/cobbler/dhcp.template (and NOT /etc/dhcp/dhcpd.conf, but they are very similar)
- Edit /etc/cobbler/settings server and next_server entries, and set manage_dhcp to 1
- “cobbler get-loaders” to get updated boot loaders
- “cobbler sync” to get settings effective. Respond to remarks.
- Check that DHCP is working (firewall?)
- Wireshark is your friend. iptables too.
My first PXE boot

- We just take any kernel and initrd and want to see that it’s being loaded
- The boot will halt since there’s no local disk

```bash
# cobbler distro add --name=justatest --kernel=/path/to/vmlinuz-2.6.27
  --initrd=/path/to/initrd-2.6.27.img
# cobbler profile add --name=testingPXE --distro=justatest
# cobbler sync
```
Giving cobbler the boot

- Cobbler will not import a kernel without an initrd
- I don’t want an initrd
- So I shut down cobblerd
- I wasn’t very sad about that
- Configuration files follow
ddns-update-style interim;

allow booting;
allow bootp;

ignore client-updates;
set vendorclass = option vendor-class-identifier;

subnet 10.1.1.0 netmask 255.255.255.0 {
    option routers 10.1.1.10;
    option domain-name-servers 10.10.0.1, 10.10.0.2;
    option subnet-mask 255.255.255.0;
    range dynamic-bootp 10.1.1.150 10.1.1.254;
    filename "/pxelinux.0";
    default-lease-time 21600;
    max-lease-time 43200;
    next-server 10.1.1.1;
}

host diskless {
    fixed-address 10.1.1.111;
}
Introduction
Setting up the filesystem
Handling the boot process
Wrapping up

/var/lib/tftpboot/pxelinux.cfg/default

DEFAULT menu
PROMPT 0
MENU TITLE Cobbler | http://fedorahosted.org/cobbler
TIMEOUT 20
TOTALTIMEOUT 600
ONTIMEOUT nfs-centos55

LABEL nfs-centos55
  kernel /images/vmlinuz-2.6.37-NFS2
  MENU LABEL nfs-centos55
  MENU DEFAULT
  append ip:::dhcp text rootfstype=nfs root=/dev/nfs nfsroot=10.1.1.1:/cent55_root
  ipappend 2

MENU end
Preparing a custom kernel

- Needs to have NFS compiled in
- Needs to have facilities to boot from NFS
- Needs to have Ethernet card’s driver compiled in
- Drivers as modules won’t cut
- Enable CONFIG_ROOT_NFS, CONFIG_NET ETHERNET, CONFIG_IP_PNP, CONFIG_IP_PNP RARP, CONFIG_IP_PNP BOOTP, CONFIG_IP_PNP DHCP
- Copy bzImage to /var/lib/tftpboot/images, and not necessarily to /boot directory as usual
Almost there...

- The originally installed system expects the root on a real hard disk.
- Fix `/etc/fstab`:

  10.1.1.1:/cent55_root / nfs defaults 0 0
A last yak to shave

- The system expects to boot with an initramfs
- Examining it shows that it does nothing...
- ... well almost nothing. It does mount /dev as a tmpfs and mknod’s the basic set of devices
- So, done once manually after chrooting to diskless’ root file system:

```bash
mkdir /dev/pts
mkdir /dev/shm
mkdir /dev/mapper
mknod /dev/null c 1 3
mknod /dev/zero c 1 5
mknod /dev/urandom c 1 9
( ... etc... )
mknod /dev/ttyS2 c 4 66
mknod /dev/ttyS3 c 4 67
chmod 0755 /dev/{null,zero,urandom,systty,tty,console,ptmx} /dev/tty*
```
That’s it!

• Admit that it sounds simple.
Wrapping up
A lot of yak shaving
Most yaks could have been avoided, had the distribution avoided closing things just because they are not in the mainline.
Lack of disk cache makes certain operations slow (compiling the kernel, anyone?)
Works smoothly once set up
Very useful and cost-effective for dedicated applications
Sources

- Documentation/filesystems/nfs/nfsroot.txt in your kernel source directory
- A nice Howto:
  http://www.faqs.org/docs/Linux-mini/NFS-Root.html
- The cobbler man page
- My own blog entry:
  http://billauer.co.il/blog/2011/01/diskless-boot-nfs-cobbler/