HOWTO Clone Disk Images on Linux Booted from a Network

Guilherme Tupynambá

gtupy (at) uol.com.br

2002–09–09

Revision History

Revision 0.3	2002–09–24	Revised by: gct
Review suggestions incorporated		
Revision 0.2	2002–09–23	Revised by: jyg
Minor revisions		
Revision 0.1	2002–09–09	Revised by: gct
First draft.		

This document describes a setup that allows a machine to boot Linux from BOOTP/TFTP, using the Grub boot loader, and save and restore disk and partition images to and from a TFTP server.

Table of Contents

1. Legal Notices......1
1.1. Disclaimer......1
1.2. Copyright......1
1.3. Feedback......1

2. Introduction......2
2.1. Why clone disk images......2
2.2. Why boot from a network......2
2.3. Network boot process overview......2

3. Setting up DHCP and TFTP servers......3
3.1. Setting up DHCP......3
3.2. Setting up TFTP......3
3.3. Using different servers......4

4. Preparing boot files......5
4.1. Kernel......5
4.2. Files on initrd......5
4.3. Packing initrd......6

5. Booting from Grub floppy disk......8
5.1. Grub menu file......8
5.2. Compiling Grub with network support......8
5.3. Making the boot floppy disk......9

6. Running the clone script......10
6.1. Saving and restoring disk images......10
6.2. Using fdisk......10

7. Extending the solution......11
7.1. Saving and restoring files instead of file systems......11
7.2. Setting up the master boot record......11
7.3. Loading necessary modules......11
7.4. Predefined operations on grub.conf......11

A. List of files on initrd......12

B. Clone script......13

References......16

1. Legal Notices

1.1. Disclaimer

This article assumes that the reader using the described setup is familiar with the technical concepts and commands. If you are not comfortable issuing the commands in this document or don't understand the clone script, don't try these instructions.

Additionally, if you do not know the difference between `/dev/hda` and `/dev/hda1`, please do not use this HOWTO. This may be the difference between restoring a partition and losing all your data. Please note that I take no responsibility for anything that may go wrong, even if the error resulting from any incorrectness in this article.

For the purpose of this document, the RedHat 7.3 distribution has been used in both server and client. Although this shouldn't make great difference, no test has been made on other distributions or custom setups.

1.2. Copyright

1.3. Feedback

Comments on this article will be highly appreciated. Direct your comments to `<gtupy (at) uol.com.br>`.

2. Introduction

2.1. Why clone disk images

The main reason to clone disk images is to ease the installation of an operating system and a basic set of applications over a large number of machines. One standard machine is prepared and its pristine image is saved to be restored on other machines saving efforts and simplifying procedures.

2.2. Why boot from a network

Booting from hard disk would limit the possibilities of copying images. It wouldn't be possible, for instance, to safely copy to and from a partition mounted by the booted operating system. Also, the operating system may not be Linux, in which case free options to clone are not available.

2.3. Network boot process overview

The client machine boots from a Grub floppy disk. Then, using the Grub BOOTP support, it gets an IP address from a DHCP server. Next, the client machine downloads the kernel and initrd images from the TFTP server. Once the initrd image is mounted in memory, the initialization script is run, making use of the programs and files stored in this image. This script allows block devices contents to be saved in the TFTP server, and contents from the TFTP server to be written to the block devices.

3. Setting up DHCP and TFTP servers

A DHCP server is required to provide IP addresses for the clients when booting Grub (BOOTP) and later when booting Linux. A TFTP server is required to make the boot images available on the network for Linux to boot. The TFTP server is also necessary to make it possible to save and restore the disk images.

3.1. Setting up DHCP

Details on DHCP are beyond the scope of this article. The "Linux Networking HOWTO" has a chapter on DHCP.

Setting up DHCP is very easy, but if you are in a network environment administered by someone else, it's advisable to use a preexisting DHCP server. If you "own" the network then you can follow this procedure.

Install DHCP, if not installed, from the rpm package, normally found in Linux distributions:

```
# rpm -ihv dhcp-*.rpm
```

Edit the `/etc/dhcpd.conf` file to configure DHCP service. In our setup, the server has IP address 10.0.0.1 and provides IP addresses up to 253 clients. Configure `/etc/dhcpd.conf` according to your environment:

```
#/etc/dhcpd.conf
server-identifier dhcp.clonedomain.com;
default-lease-time      172800;
max-lease-time          604800;
option domain-name      "clonedomain.com";
subnet 10.0.0.0 netmask 255.255.255.0 {
        range dynamic-bootp 10.0.0.2 10.0.0.254;
}
```

Start the dhcpd server:

/etc/rc.d/init.d/dhcpd start.

3.2. Setting up TFTP

Setting up TFTP is almost as easy as DHCP.

First install from the rpm package:

```
# rpm -ihv tftp-server-*.rpm
```

Create a directory for the files:

```
# mkdir /tftpboot
# chown nobody:nobody /tftpboot
```

The directory `/tftpboot` is owned by user `nobody`, because this is the default user id set up by tftpd to access the files.

Edit the file `/etc/xinetd.d/tftp` to look like the following:

```
service tftp
{
        socket_type     = dgram
        protocol        = udp
        wait            = yes
        user            = root
        server          = /usr/sbin/in.tftpd
        server_args     = -c -s /tftpboot
        disable         = no
        per_source      = 11
        cps             = 100 2
}
```

The changes from the default file are the parameter *`disable = no`* (to enable the service) and the server argument *`-c`*. This argument allows for the creation of files, which is necessary if you want to save boot or disk images. You may want to make TFTP read only in normal operation.

Then reload xinetd:

/etc/rc.d/init.d/xinetd reload

You can use the **tftp** command, available from the tftp (client) rpm package, to test the server. At the tftp prompt, you can issue the commands **put** and **get**.

3.3. Using different servers

It is possible to use different servers for DHCP and TFTP. This may be necessary if using a preexisting DHCP server. You can configure the *`next-server`* parameter in DHCP to point to the TFTP server or you can use the command **tftpserver** in Grub.

4. Preparing boot files

Now that the server is set up, you need to prepare the files to make the client boot. Two files are necessary: the kernel and the init ramdisk (initrd) which will be mounted by the kernel as the root file system. This document assumes that the procedures outlined in this section and the next are made in the client machine. Normally, when saving and restoring disk images, there is no need to have Linux installed on a local hard disk. To deploy disk images to a number of machines, start by installing a Linux distribution on one machine for each model. Use DHCP and have TFTP client to test the setup made in the previous section. Unless otherwise noted, commands are issued in the **bash** shell by the user `root` in a working directory.

4.1. Kernel

Identify the compressed kernel file:

```
# cd /boot
# ls vmlinuz-$(uname -r)
vmlinuz-2.4.18-3
```

The version may vary, according to your system. Upload this file to the TFTP server, renaming it to `vmlinuz`:

```
# tftp 10.0.0.1
tftp> binary
tftp> put vmlinuz-2.4.18-3 vmlinuz
Sent 1030147 bytes in 2.3 seconds
tftp> quit
```

You may want to keep the file name, or have different names for different hardware models or kernel versions. We use the same names for the kernel and initrd so we don't have to make another boot floppy disk after changing the kernel or the initrd images.

4.2. Files on initrd

Next, make the root file system image for the client. The full listing of the files is in Appendix A.

These files have been taken from a working system as a minimum configuration for having powerful shell (**bash**), client network utilities (**dhcpcd** and **tftp**), and copying and compressing utilities (**dd**, **gzip**). Administrative commands (**mknod**, **mount**, **fdisk** and **insmod**) are also present.

In the working directory create a file named `initrd.lst` and put these file names on it. To check the existence of these files in your system, run the following command:

```
# ls -d $(<initrd.lst) > /dev/null
```

You should get an error output like this:

```
ls: /bin/clone: No such file or directory
```

```
ls: /bin/tftp: No such file or directory
ls: /lib/3c59x.o: No such file or directory
```

The first error is a script to be created in the working directory. The second error is the program **tftp** found in the directory `/usr/bin` instead of `/bin`. The third is the network interface card module (probably not yours) found in the directory `/lib/modules/$(uname -r)/kernel/drivers/net`.

These three files will be discussed in upcoming sections separately soon. If there are other missing files, check for lack of installation or differences in version, distribution or hardware. Adjust the list to match your system.

4.3. Packing initrd

The next step is to make the image. A size of 4 Mbytes was enough to hold the files in our setup. You may increase this size if necessary.

```
# dd if=/dev/zero of=initrd bs=1024 count=4096
4096+0 records in
4096+0 records out
# yes | mkfs initrd
mke2fs 1.27 (8-Mar-2002)
initrd is not a block special device.
Proceed anyway? (y,n) Filesystem label=
blah blah blah...
# mkdir mnt
# mount -o loop initrd mnt/
# egrep -v "clone|3c59x|tftp" initrd.lst | cpio -pdm mnt
4876 blocks
```

Now the three files excluded by the **egrep** command. Copy the `tftp` program from `/usr/bin` to the image directory:

```
# cp -p /usr/bin/tftp mnt/bin/
```

Identify the proper module for your network interface card. Use the output of the commands **lspci** and **lsmod** to identify the file, which resides in the directory `/lib/modules/$(uname -r)/kernel/drivers/net`.

Whenever you see a reference to *3c59x*, use the name of the module suited for your case.

```
# cp -p /lib/modules/$(uname -r)/kernel/drivers/net/3c59x.o mnt/lib/
```

Edit the clone script found in Appendix B, changing the variables as explained in Section 6. Make it executable and copy it to the image directory:

```
# chmod +x clone
# cp -p clone mnt/bin/
```

Unmount, compress, and send the initrd image.

```
# umount mnt/
# gzip initrd
# tftp 10.0.0.1
tftp> binary
tftp> put initrd.gz
Sent 1155530 bytes in 2.8 seconds
tftp> quit
```

5. Booting from Grub floppy disk

The next step is to make a boot floppy disk using Grub. GNU Grub is the GRand Unified Bootloader. It can handle BOOTP and TFTP, so it can boot from network.

5.1. Grub menu file

In the working directory create a file named `grub.conf` with the following content:

```
default=0
timeout=1
title Clone
        bootp
        root (nd)
        kernel /vmlinuz rw root=/dev/ram ramdisk_size=4096 init=/bin/clone
        initrd /initrd.gz
```

In the last four lines are the Grub commands to boot from network:

- **bootp**, to get an IP address from the DHCP server.
- **root (nd)**, to set the root in the network (TFTP server). An alternative TFTP server could be set before this command using the command **tftpserver** <*`tftp server`*>.
- **kernel**, to specify the kernel file and its parameters:
  - *`rw`*, to specify writable mounting of the root file system.
  - *`root`*, to specify where to mount the root file system (in ram memory).
  - *`ramdisk_size`*, to specify the ram disk size. 4096 (kbytes) is the default size but if you needed a greater image, change this parameter accordingly.
  - *`init`*, to specify (our script) as the first program to run in user mode (in the absence of **init** and **sh**).
- **initrd** to specify the file holding the image of the root file system.

5.2. Compiling Grub with network support

To compile Grub, first download the source tarball from the Grub web site and unpack it. Run **configure** specifying the menu file you just created and the network interface card model. Run **make** as usual.

```
# tar xzf grub-0.92.tar.gz
# cd grub-0.92
# ./configure --enable-preset-menu=../grub.conf --enable-3c90x
# make
```

Again, where you see *3c90x* put the model of your network interface card. First check if it is supported by Grub.

5.3. Making the boot floppy disk

Once Grub is compiled, the image of the boot floppy disk is the concatenation of the files `stage1/stage1` and `stage2/stage2`. To make the floppy disk run:

```
# cat stage1/stage1 stage2/stage2 | dd of=/dev/fd0
```

You should now have a boot floppy disk.

6. Running the clone script

The clone script, shown in Appendix B, is not essential. You can make *init=/bin/bash* as a kernel parameter and end up with a shell from where you can run the available commands and programs.

The script is presented here to show the commands in a formal way and to propose a way to reduce the possibility of damages resulting from mistyping. You have to change the variables `tftp_server`, `nic_module`, `major_a`, `family_a` and `image_a` according to your environment and application.

Please note that the arrays `major_a` and `family_a` are corresponding. Wrong major number for a given family name will mislead the user. You can locate the major and minor numbers of the devices of interest (whole disks and partitions) by listing the `/dev` directory. The major and minor number are where the size of a regular file is, in the output of the command **ls −l**, separated by a comma.

```
# ls -l /dev/fd0 /dev/hda /dev/hda1 /dev/hdc
brw-rw----    1 root     disk       2,   0 Apr 11 11:25 /dev/fd0
brw-rw----    1 root     disk       3,   0 Apr 11 11:25 /dev/hda
brw-rw----    1 root     disk       3,   1 Apr 11 11:25 /dev/hda1
brw-rw----    1 root     disk      22,   0 Apr 11 11:25 /dev/hdc
```

The command **set −e** instructs the shell to abort the script should any command return non−zero code. The message"Kernel panic: Attempted to kill init!" will follow, as in case of normal end. Don't panic! This is normal, given the circumstances. Just turn off the computer. Press Ctrl−Alt−Del to have a smooth reboot *before* exiting the script to avoid this ugly message.

The command **insmod** will load the network interface module and the command **dhcpcd** will start DHCP client. Note that the fact that Grub used DHCP during its boot has nothing to do with Linux doing the same.

The script makes a big loop and, for each iteration, it asks for one of three operations: **Copy from network to device**, **Copy from device to network** or **Run fdisk**. Then the script asks which block device to use. The array `major_a` holds the major number for the block devices allowed to be used and the array `family_a` the respective names for the device families. Next, the script asks the minor number of the block device to be used.

6.1. Saving and restoring disk images

If one of the copy operations is selected, the script asks for the name of the image to be saved or restored. The image name is limited to the elements of the array `image_a`. A named pipe with the same name as the image is created if doesn't exist. Finally, **dd** and **tftp** are invoked simultaneously to transfer the image. Unlike regular **ftp**, **tftp** puts and gets named pipes just like regular files.

6.2. Using fdisk

If **fdisk** command is selected it is invoked for the block device. **fdisk** is normally run for the whole disk as opposed to one partition. Note that what normally is called `/dev/hda` will be called `/dev/hda0` by the clone script. The input of **fdisk** could be put in the script to make automatic creation of partitions if desired.

7. Extending the solution

7.1. Saving and restoring files instead of file systems

If you don't want to save a whole disk image just the files within the file system, you can use a similar solution but with **tar** or **cpio** instead of **dd**. Also you need to mount the file system. More commands should be added to the clone script as shown below.

```
# mkdir /mnt
# mount ${device_name} /mnt
# mknod ${image} p
# tftp ${tftp_server} <<-EOT &
binary
put ${image}
EOT
tar czf ${image}
```

or

```
# tftp ${tftp_server} <<-EOT &
binary
get ${image}
EOT
tar xzf ${image}
```

You have to put the **mkdir** and **tar** programs in the initrd image so that the script can use them.

7.2. Setting up the master boot record

In a situation where you used this setup to reorganize and resize your partitions, you may end up with a disk that doesn't boot. Running the **setup** command from Grub (including the **grub** program in the image) should resolve the situation. See the Grub documentation for details.

7.3. Loading necessary modules

Depending on your kernel, additional modules may be necessary to access some block devices like SCSI devices. Just put the necessary modules in the `/lib` directory of the initrd image and the correspondent **insmod** commands in the clone script. The same applies for file systems. If, for instance, you want save the files instead of the image of a *fat* file system you will need the `fat.o` and `vfat.o` modules.

7.4. Predefined operations on `grub.conf`

The Grub menu file `grub.conf` may be customized to present a few copy options or even execute a predefined operation such as repartitioning the disk and retrieving specified images from network. Again, you can use the concepts presented here to achieve a specific application.

A. List of files on initrd

```
/bin/
/bin/bash
/bin/clone
/bin/dd
/bin/gzip
/bin/mknod
/bin/mount
/bin/tftp
/dev/
/dev/console
/dev/null
/etc/
/etc/dhcpc/
/etc/hosts
/etc/nsswitch.conf
/etc/protocols
/etc/services
/lib/
/lib/3c59x.o
/lib/i686/
/lib/i686/libc-2.2.5.so
/lib/i686/libc.so.6
/lib/ld-2.2.5.so
/lib/ld-linux.so.2
/lib/libdl-2.2.5.so
/lib/libdl.so.2
/lib/libnss_files-2.2.5.so
/lib/libnss_files.so.2
/lib/libtermcap.so.2
/lib/libtermcap.so.2.0.8
/proc/
/sbin/
/sbin/dhcpcd
/sbin/fdisk
/sbin/insmod
/tmp/
/var/
/var/run/
```

B. Clone script

```
#!/bin/bash

set -e

export PATH=/sbin:/bin

tftp_server=10.0.0.1
nic_module=3c59x.o
major_a=(2 3 22)
family_a=(fd hda hdc)
image_a=(img0001 img0002 img0003 img0004)

operation_a=(   "Copy from network to device" \
                "Copy from device to network" \
                "Run fdisk")

mount -t proc proc /proc
insmod /lib/${nic_module}
/sbin/dhcpcd

while true; do \
        [ ! -z "${image}" ] && unset image
        echo
        echo "Clone Menu"
        echo
        echo "Operation"
        echo
        PS3="Choose operation (1-${#operation_a[*]}): "
        select operation in "${operation_a[@]}"; do \
                [ -z "${operation}" ] && continue
                echo
                echo $REPLY - $operation
                echo
                break
        done

        echo "Device Family"
        echo
        PS3="Choose device family (1-${#family_a[*]}): "
        select family in "${family_a[@]}"; do \
                [ -z "${family}" ] && continue
                echo
                echo $REPLY - $family
                echo
                break
        done

        major_i=$[REPLY-1]
        major=${major_a[$major_i]}

        echo "Minor Number"
        echo
        PS3="Choose minor number (0-255): "
        echo -n "$PS3" >&2
        read minor
        minor=$[minor%256]
        echo
        echo $minor
        echo
```

```
if [ "${operation}" != "${operation_a[2]}" ]; then \
        echo "Image"
        echo
        PS3="Choose image (1-${#image_a[*]}): "
        select image in "${image_a[@]}"; do \
                [ -z "${image}" ] && continue
                echo
                echo $REPLY - $image
                echo
                break
        done
        image_i=$[REPLY-1]
        image=${image_a[${image_i}]}
fi

echo
echo -e "Operation:\t$operation"
device_name=/dev/${family_a[${major_i}]}${minor}
echo -e "Device:\t\t${device_name} ($major, $minor)"
[ ! -z "${image}" ] && echo -e "Image:\t\t${image}"
echo

echo "Confirmation"
echo
PS3="Ok/Cancel (1-2): "
select ok in Ok Cancel; do \
        [ -z "${ok}" ] && continue
        echo
        echo $REPLY - $ok
        echo
        break
done
if [ "${ok}" = "Ok" ]; then \
        if [ ! -b ${device_name} ]; then \
                echo "Creating ${device_name}"
                mknod ${device_name} b ${major} ${minor}
        fi
        if [ ! -z "${image}" ]; then \
                if [ ! -p ${image} ]; then \
                        echo "Creating pipe"
                        mknod ${image} p
                fi
        fi
        if [ "${operation}" = "${operation_a[0]}" ]; then \
                tftp ${tftp_server} <<-EOT &
                binary
                get ${image}
                EOT
                gzip -c -d < ${image} | dd of=${device_name}
        elif [ "${operation}" = "${operation_a[1]}" ]; then \
                tftp ${tftp_server} <<-EOT &
                binary
                put ${image}
                EOT
                dd if=${device_name} | gzip -c > ${image}
        elif [ "${operation}" = "${operation_a[2]}" ]; then \
                fdisk ${device_name}
        fi
        echo
fi
```

```
        echo "Continuation"
        echo
        PS3="Continue/Exit (1-2): "
        select new in Continue Exit; do \
                [ -z "${new}" ] && continue
                echo
                echo $REPLY - $new
                echo
                break
        done
        [ "${new}" = "Exit" ] && break
done
exit 0
```

References

[grub] *GRUB*.

http://www.gnu.org/software/grub/