NFS−Root−Client Mini−HOWTO

Ofer Maor

v4.1, 02 Feb, 1999

Revision History

Revision 4.1	Feb 02, 1999	Revised by: mo

The purpose of this Mini−Howto is to explain how to create client root directories on a server that is using NFS Root mounted clients.

Table of Contents

1. Copyright

If you have questions, please contact Ofer Maor (<oferm@hcs.co.il>), the author of this mini–HOWTO, or Greg Hankins, the Linux HOWTO coordinator, at <gregh@sunsite.unc.edu> via email, or at +1 404 853 9989.

If you have anything to add to this Mini–Howto, please mail the author (Ofer Maor, <oferm@hcs.co.il>), with the information. Any new relevant information would be appreciated.

1.1. Thanks

I would like to express my thanks to the author of the NFS–Root Howto, Andreas Kostyrca (<andreas@medman.ag.or.at>). His Mini–Howto helped me with the first steps in creating a NFS Root Mounted client. My Mini–Howto does not, in any way, try to replace his work, but to enhance it using my experiences in this process.

I would also like to thank Mark Kushinsky (<mark026@ibm.net>) for polishing the english and spelling of this Howto, thus making it much more readable.

2. Preface

This Mini−Howto was written in order to help people who want to use NFS Root mounting to create their client's directories. Please note that there are many ways to accomplish this, depending on your needs and intent. If the clients are individual, and each client has its own users and administrator, it will be necessary to make significant parts of the client dirs not shared with other clients. On the other hand, if the client is intended for multiple users, and are all administrated by the same person (for instance, a computerclass), make as many files as possible shareable in order to make administration more manageable. This Howto will focus on the second issue.

2.1. General Overview

When building a client's root directory, and trying to limit ourselves to the minimum client size, we mainly focus on which files we can share, or mount from the server. In this Howto I will recommend the configuration of a client based on my experience. But beforewe begin please note:

- This Mini−Howto does not explain how to do the actual NFS Root mounting. Refer to the NFS−Root Mini−Howto if you need more information about that issue.
- I based most of my client's configuration on mounts and symbolic links. A lot of those symbolic links can be replaced by hardlinks. One should choose according to his personal preference. Putting a hardlink over a mount and a symbolic link has its advantages, but might cause confusion. A file will not be erased until all its hardlinks are removed. Thus, In order to prevent a case in which you upgrade a certain file, and the hardlinks still refer to the older version, you will have to be very careful and keep track of every link you put.
- While mounting the information from the server, two concepts can be used. The first (most common) concept, is to mount the whole server root directory under a local directory, and then just change the path or link the relevant directories there. I personally dislike mounting root partitions of a server on clients. Thus, this Howto suggests a way to mount the relevant directories of the server to the relevant places on the system.
- This Howto is based on my experience building client directories on a Slackware 3.1 based distribution. Things may be different (especially on the `rc.*` files), for other users, however the concepts should still remain the same.

3. Creating the client's root directory

3.1. Creating the directory tree

First of all, you need to create the directory structure itself. I created all the clients under `/clients/hostname` and I will use it for my examples listed below. This, however, can be changed to anything else. The first stage, then, is to create the relevant directories in the root directory. You should create the following directories:

`bin` , `dev` , `etc` , `home` , `lib` , `mnt` , `proc` , `sbin` , `server` , `tmp` , `usr` , `var`

and any other directories you might want to have on your system.

The `local`, `proc`, and `dev` directories will be used separately on each machine while the rest of the directories will be either partly or completely shared with the rest of the clients.

3.2. Creating the minimal file system needed for boot

3.2.1. Creating the `dev` dir.

Although the `dev` dir can be shared, it is better to create a separate one for each client. You can create your client's `dev` directory with the appropriate MAKEDEV scripts, however in most cases it is simpler just to copy it from the server:

```
bash# cp -a /dev /clients/hostname
```

You should keep in mind that `/dev/mouse`, `/dev/cdrom` and `/dev/modem` are symbolic links to actually devices, and therefore you should be sure that they are linked correctly to fit the client's hardware.

3.2.2. Copying the necessary binaries.

Although we mount everything from the server, there is a minimum that we need to copy to each client. First of all, we need "init", our system will not be able to run anything before init'ing (as the author found out in the hard way ;–). So first, you should copy `/sbin/init` to your client's `sbin` dir and then so that `rc.S` will run, you should copy `/bin/sh` to the client's `bin` directory. Also, in order to mount everything you need to copy `/sbin/mount` to the client's `sbin` directory. This is the minimum, assuming the first line in your `rc.S` is **mount –av**. However, I recommend copying a few more files: `update`, `ls`, `rm`, `cp` and `umount`, so that you will have the basic tools in case the client has problems mounting. If you choose to leave your swap on line before mount, you should also copy the `swapon` binary.

Since most of these binaries are by default dynamically linked, you will also need to copy a fair part of `/lib`:

```
bash# cp -a /lib/ld* /lib/libc.* /lib/libcursses.* /client/hostname/lib
```

Hardlinking the binaries themselves, instead of copying them, should be considered. Please read my comments on this in Section 2.1 of this Howto.

Please notice, all of the information above assumes that the kernel has been given the network parameters while booting up. If you plan to use rarp or bootp, you will probably need the relevant binaries for these as well.

Generally, you will need the minimum of files that will enable you to configure the network and run `rc.S` up to the point where it mounts the rest of the file system. Make sure you looked into your `/etc/init` and `rc.S` files, making sure there are no "surprises" in any of them, which will require other files to be accessed, before the first mount will take place. If you do, however, find such files, you can either copy them as well, or remove the relevant parts from your `init` and your `rc.S` files.

3.2.3. The `var` directory

The var directory, in most cases, should be separate for each client. However, a lot of the data can be shared. Create under the server directory a directory called `var`. We will mount the server's `var` directory there. To create the local `var` directory, simply type:

```
bash# cp −a /var /clients/hostname/
```

Now, you have a choice as to what you want to separate, and what you want to share. Any directory/file that you want to share, simply remove it from the client's `var` dir, and symlink it to the `/server/var/` directory. However please note that you should either symlink it to `/server/var` or to `../server/var` but NOT to `/clients/hostname/server/var` as this will not work when the root changes.

Generally, I would recommend separating `/var/run`, `/var/lock`, `/var/spool`, and `/var/log`.

3.2.4. The rest of the directories

- `etc` is explained thoroughly in the next section.
- `mnt` and `proc` are for local purposes.
- `usr` and `home` are merely mount points.
- `tmp` is up to you. You can create a different `tmp` directory for each client, or create some `/clients/tmp` directories, and mount it for each client under `/tmp`. I would recommend that you provide each client with a separate `tmp` directory.

3.3. Building the `etc` directory and configuring the clients

Please Note −

> **Note:** this section refers to building the etc directory which is mostly shared among the clients. If your diskless clients have separate system administrators, it's best to set up for each client a separate etc directory.

3.3.1. Building a clients wide `etc` directory

3.3.2. Creating a client's `etc` directory

Although we separate the `etc` directories of the clients, we still want to share a large portion of the files there. Generally, I think sharing the etc files with the server's `/etc` is a bad idea, and therefore I recommend creating a `/clients/etc` directory, which will hold the information needed for the clients. To start with, simply copy the contents of the server's `etc` to the `/clients/etc` directory.

You should add to this directory all of the non−machine−specific configuration files, for instance `motd`, `issue`, etc. and not the clientspecific ones.(i.e. `inittab` or `fstab`)

The most important changes will be in your `rc.d` directory. First, you should change `rc.inet1` to be suitable for your local setup. I pass all my network parameters to the kernel through the LILO/Loadlin, therefore I remove almost everything from `rc.inet1` file. The only thing I leave there is the **ifconfig** and **route** of the localhost. If you use rarp or bootp, you will have to build it accordingly.

Secondly, you should edit your `rc.S`. First, remove all the parts that are responsible for the fsck check as fsck will occur when the server boots up. Then, you should find the line that mounts your fstab. This should look something like:

mount −avt nonfs

The *−t nonfs* is there since normal clients first run `rc.S` and only later on use `rc.inet1` to configure the Ethernet. As this will cause no NFS partitions to be mounted this line should be deleted. Therefore, change it to *mount −av*. If you need to run rarp/bootp to configure your network, do it in `rc.S` (or call the appropriate script from `rc.S`), before the mount, and make sure your physical `bin` and `sbin` directories have the necessary files available.

After the **mount −av** is performed, you will have a working file system. Build a general fstab, so that you can later copy it to each client. Your fstab should look something like this:

Table 1. fstab

server:/clients/hostname	/	nfs	default 1 1
server:/bin	/bin	nfs	default 1 1
server:/usr	/usr	nfs	default 1 1
server:/sbin	/sbin	nfs	default 1 1
erver:/home	/home	nfs	default 1 1
server:/lib	/lib	nfs	default 1 1
server:/clients/etc	/server/etc	nfs	default 1 1
server:/clients/var	/server/var	nfs	default 1 1
none	/proc	proc	default 1 1

Please notice, that the keyword *default* might not work on all versions of mount. You might change it to `rw` or `ro` or remove all of the `default 1 1` part.

Also, make sure your server's `/etc/exports` looks like this:

Table 2. /etc/exports

/clients/hostname	hostname.domainname(rw,no_root_squash)
/clients/etc	hostname.domainname(ro,no_root_squash)
/clients/var	hostname.domainname(ro,no_root_squash)
/usr	hostname.domainname(ro,no_root_squash)
/sbin	hostname.domainname(ro,no_root_squash)
/bin	hostname.domainname(ro,no_root_squash)
/lib	hostname.domainname(ro,no_root_squash)
/home	hostname.domainname(rw,no_root_squash)

Other than the first line, which should be separate for each host, the rest of the lines can be replaced with a hostmask to fit all your hosts (like pc*.domain − keep in mind though, that * will substitue only strings without a dot in them). I suggest that you make most of the directories read only, but this is up to you. The `no_root_squash` will make sure root users on the clients have actual root permissions on the nfsd as well. Check out man exports(5). If you want users to be able to run **passwd** from the clients also, make sure the `/etc` has `rw` and not `ro` permissions. However, this is not advisable.

Please note another thing concerning the `rc.S` file. In Slackware, by default, it creates a new `/etc/issue` and `/etc/motd` every time it runs. This function MUST be disabled if these files are mounted ro from the server, and I would recommend that they should be disabled in any case.

Lastly, if you want to have the same userbase on the server as on the clients, you should choose between 1), using NIS (Yellow Pages − check the yp−howto), and then each client will have a separate `/etc/passwd` and `/etc/group` as it receives it from the NIS server. 2) In most cases, a simple symbolic link will suffice. Therefore, you will need to either hardlink `/clients/etc/passwd` to `/etc/passwd`, or if you prefer a symlink, link `/etc/passwd` to `/clients/etc/passwd` (and not the other way around, since the clients do not mount the server's `etc` directory). Do the same for `/etc/group`.

3.3.3. Creating a client's `etc` directory

Generally, most of the files in the client's etc should be symlinked to the `/server/etc` directory. However, some files are different for each machine, and some just have to be there when the kernel loads. The minimum you need from the `etc` dir is as follows:

```
resolv.conf
hosts
inittab
rc.d/rc.S
fstab
```

Since these 5 files can be identical on all clients, you can simply hardlink them or copy them again. However, with the `rc.S` and `fstab` file it is advised to keep a separate copy for each client. You will also need a separate `etc/HOSTNAME` for each client. I personally recommend having all of the `rc.d` files separate for each client, as configuration and hardware might vary from one to another.

For each client, add to the `fstab` the proper swap line:

Table 3. fstab

/dev/swap_partition swap swap default 1 1

The rest of the `/etc` files of the client, you can either hardlink to the `/clients/etc/*` files, or symlink them to the `/server/etc` (which is the mount point of `/clients/etc/`).

Make sure your machine can resolve properly, either through a named or through `etc/hosts`. It is not a bad idea to keep the server's IP in the `etc/hosts`, instead of counting on resolving. If you will count only on named resolving, a problem in the named will prevent your clients from booting up.

3.4. Booting Up

Now, all you have to do is to boot up your machine, cross your fingers and hope everything works as it should :−).

4. Creating more clients

If you have followed my instructions so far this should be simple – cd to `/clients/` and type:

```
bash# cp -a hostname1 hostname2
```

and then make sure you check these points:

- `rc.d/*` files matches the hardware and wanted software configuration
- `etc/HOSTNAME` is correct
- fstab's swap line is correct
- the symbolic links of `dev/mouse`, `dev/modem` and `dev/cdrom`are right.

Good Luck....