Root over NFS – Another Approach
# Table of Contents

**Root over NFS – Another Approach**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root over NFS – Another Approach</td>
<td>1</td>
</tr>
<tr>
<td>George Gousios, <a href="mailto:cs98011@icsd.aegean.gr">cs98011@icsd.aegean.gr</a></td>
<td>1</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Setting up the server</td>
<td>1</td>
</tr>
<tr>
<td>3. Setting up the clients</td>
<td>1</td>
</tr>
<tr>
<td>4. Preparing the boot disk</td>
<td>1</td>
</tr>
<tr>
<td>5. The magic time</td>
<td>1</td>
</tr>
<tr>
<td>6. Other Stuff</td>
<td>1</td>
</tr>
<tr>
<td>7. Appendix</td>
<td>2</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>2</td>
</tr>
<tr>
<td>1.1 The setting</td>
<td>2</td>
</tr>
<tr>
<td>1.2 The alternatives</td>
<td>3</td>
</tr>
<tr>
<td>1.3 General Principles</td>
<td>3</td>
</tr>
<tr>
<td>2. Setting up the server</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Setting up the NFS server</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Setting up the DHCP/BOOTP server</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Preparing the base system</td>
<td>6</td>
</tr>
<tr>
<td>3. Setting up the clients</td>
<td>6</td>
</tr>
<tr>
<td>3.1 Errata</td>
<td>6</td>
</tr>
<tr>
<td>3.2 Fiddling with scripts and files</td>
<td>7</td>
</tr>
<tr>
<td>How to setup a swap partition</td>
<td>7</td>
</tr>
<tr>
<td>Modifying /etc/fstab</td>
<td>8</td>
</tr>
<tr>
<td>Copying password files</td>
<td>8</td>
</tr>
<tr>
<td>3.3 Booting the base system</td>
<td>8</td>
</tr>
<tr>
<td>3.4 Configuring the system</td>
<td>8</td>
</tr>
<tr>
<td>Configuring the language</td>
<td>9</td>
</tr>
<tr>
<td>The X window system</td>
<td>9</td>
</tr>
<tr>
<td>Configuring network access for KDE2</td>
<td>10</td>
</tr>
<tr>
<td>4. Preparing the boot disk</td>
<td>11</td>
</tr>
<tr>
<td>4.1 Building a kernel</td>
<td>11</td>
</tr>
<tr>
<td>4.2 Creating the boot disk</td>
<td>11</td>
</tr>
<tr>
<td>4.3 The kernel command line</td>
<td>12</td>
</tr>
<tr>
<td>5. The magic time</td>
<td>12</td>
</tr>
<tr>
<td>6. Other Stuff</td>
<td>13</td>
</tr>
<tr>
<td>6.1 Contributors</td>
<td>13</td>
</tr>
<tr>
<td>6.2 Copyrights</td>
<td>13</td>
</tr>
<tr>
<td>6.3 Contacting the author</td>
<td>13</td>
</tr>
<tr>
<td>6.4 Changelog</td>
<td>13</td>
</tr>
<tr>
<td>7. Appendix</td>
<td>13</td>
</tr>
<tr>
<td>7.1 Appendix A – A script for creating host directories</td>
<td>13</td>
</tr>
<tr>
<td>7.2 Appendix B – A script to create the dhcpcd.conf file using arpwatch</td>
<td>15</td>
</tr>
<tr>
<td>The arp.dat2dhcpcd.conf script</td>
<td>15</td>
</tr>
<tr>
<td>7.3 Appendix C – A sample XF86Config file</td>
<td>16</td>
</tr>
</tbody>
</table>
This HOWTO does not intend to replace the existing Root over NFS Howto's. It is just another approach, particularly useful in large system installations. It is the result of many days of trying to setup a system for the University of the Aegean computer labs. The installation method described here is up and running. The HOWTO is dedicated to all of those guys who programmed these exceptionally good OS and tools. Also dedicated to all people that encouraged me to write it.

1. **Introduction**
   - 1.1 The setting
   - 1.2 The alternatives
   - 1.3 General Principles

2. **Setting up the server**
   - 2.1 Setting up the NFS server
   - 2.2 Setting up the DHCP/BOOTP server
   - 2.3 Preparing the base system

3. **Setting up the clients**
   - 3.1 Errata
   - 3.2 Fiddling with scripts and files!
   - 3.3 Booting the base system
   - 3.4 Configuring the system

4. **Preparing the boot disk**
   - 4.1 Building a kernel
   - 4.2 Creating the boot disk
   - 4.3 The kernel command line

5. **The magic time**

6. **Other Stuff**
   - 6.1 Contributors
   - 6.2 Copyrights
   - 6.3 Contacting the author
This document does not resemble a common HOWTO, meaning referencing to general principles, but it is rather an on-hand approach to a by nature complex matter. It borrows the structure of the current Root over NFS, but differs from it in the following points:

- It provides a working solution from the distribution used. The distribution specific points should be applicable to all major distributions (RedHat, SuSE, Debian).
- It uses more up to date tools, ex NFS v3.0, kernel 2.4.0, dhcp instead of bootparamd.
- All steps are described in detail, letting the reader to adapt them to his own system. No scripts!

This HOWTO expects that you have a general knowledge of what you are up to, so first read the Diskless Nodes HOW-TO.

1.1 The setting

It is a common case a University computer lab to have a lot PC's running Windows 98 or/and NT and a powerful UNIX server to satisfy the need of an alternative operating environment. This UNIX server is most of times idle or merely accessed by telnet and running stupid tasks. On the other hand, students, especially those attending a computer science department, feel like taking full advantage of it, just for fun or for "educational purposes" (breaking in, hacking it...). The restrictive environment of telnet does not allow us to enjoy the use of a power server. There are 2 alternatives to that:

- Try to persuade the department's headmaster to approve of the purchase of a bunch of new Unix workstations.
- Try to persuade the same guy to approve of transforming the server to a diskless node server.

The network at the computer lab consists of the following.

- UNIX server: SUN Enterprise 3500 with 2 64 bit SPARC@366 Mhz processors and 512 MB of memory. A real monster, isn't it?
- "Dumb" target workstations: 60–70 PC's with variable configurations, ranging from PII 266 to PIII 450 with 64–128 MB RAM.

The task I had to accomplish was the following: Provide a complete working solution without new expenses and without modifying anything but the necessary on the server.
1.2 The alternatives

Being the responsible for the project, I had to choose between a variety of solutions about it. I choose the following, for the reasons illustrated:

- The new 2.4 kernel: It provides a robust and fast solution, using less memory than the old 2.2 series. If it is important for your users to attach devices to their PC's then it is the only solution. Also provides NFS v3, and more efficient memory management.
- The KDE 2.1.1 desktop environment: VERY stable, easy to use, Internet enabled, makes the transition from Windows to Linux desktop almost effortless. GNOME + Afterstep is another option, but not as mature as a solution as KDE.
- SuSE 7.0 distribution: My favorite one, IMHO the most balanced between ease of use and understanding of a Linux system structure.

1.3 General Principles

To be able to boot a Linux system, you have to provide it with the following:

- The /sbin directory. There exists the init programm, which is responsible for starting other programms and start up scripts during the boot process. Also, the /sbin directory contains the startup scripts in the case of SuSE, some useful programms like the portmap programm and many other programms that are needed before you mount the /usr directory.
- The /lib directory. It contains the libc libraries that are absolutely necessary if your init is dynamically linked.
- The /bin directory. It contains file commands and shells for running startup scripts.
- The /etc directory. It contains configuration files for most programms and the rc.d directories that is the default for startup scripts.
- The /var directory. It is a spool area for programms that want to write somewhere. It is divided into many subdirectories with alternate usability.
- The /dev directory. It contains character and block special devices that allow programms to communicate with the computers devices via the kernel.

You should notice that after a clean install, the total size of these directories is not that big, ranging from 30 to 40 MB. The main load of files exists in the /usr and /opt directories. So, it is possible to create a directory for every diskless client containing the above listed directories and mount points for directories like /usr that will be exported by the server. The boot process, as assumed by this document, is the following:

1. The user reboots the computer, and using a diskette boots the Linux kernel.
2. The kernel takes control of the system, identifies the system devices, and uses BOOTP to obtain the IP address matching the NIC ’s hardware address.
3. The init programm is started. Before switching to a run level, it calls a script described in the /etc/inittab file. This script is responsible for building the library cache, initialise and mount a swap file, load some system specific kernel modules and set the hostname.
4. The boot script finishes and the init programm switches to the specified runlevel. It starts to execute the scripts located into the /etc/rc.d/rcX directory where 'X' is the name of the runlevel. These scripts are responsible for starting the portmapper and mounting the NFS exported /usr, /home and /opt directories.
5. The user is able to login.

To sum up, the system administrator has to do the following tasks:

- Prepare a clean install of the system to be exported to the diskless hosts.
- Create the host specific directories
- Control what is going to be started during the diskless clients' boot process.
Prepare the server to export some directories and start a bootp service.

2. **Setting up the server**

The first, and less tricky, thing to do is to setup the server. The server must be prepared to run these services:

- NFS, preferably version 3, for exporting the following directories: /usr, /lib/modules, /opt (at least at SuSE) and /home (unless you have a dedicated file server).
- DHCP server (in bootp mode), for matching the clients’ MAC addresses to IP addresses.

Also, the administrator has to create directories for each client, containing necessary startup files and programs. The directory scheme created for the installation described was like this one:

```
/usr/local/linux-
  /base-
    /bin
    /sbin
    /etc
  /workstations-
    195.251.160.100
      /bin
      /sbin
      /etc
    195.251.160.101
    195.251.160.102
    base(symbolic link to ../base)
```

The /base directory contains the whole file system you want to export to your clients. The per IP directories contain files that are needed before mounting the /usr or /lib/modules directories, like the /etc folder. This is a comfortable directory structure for 2 purposes: i) You can easily create a basic system at the base directory and copy the per workstation files at the workstation directories easily, with an entry level bash script ii) You can easily add or delete or update workstations by modifying the directories under /workstations. A script for copying the appropriate files (which will be discussed later) can be found in Appendix A.

### 2.1 Setting up the NFS server

An NFS server can be set up in two ways:

- Using the /etc/exports file at BSD–compliant Unices like Linux of FreeBSD.
- Using the /etc/dfs/dfstab at SysV Unices like Solaris.

**/etc/exports:** The /etc/exports file controls the directories to be exported and the export options per workstation. It has a structure like the following (Linux):

```
/path/to/dir1  ws1(options) ws2(options)....
/path/to/dir2  ws3(options) ws1(options)....
```
Options include ro or rw, root_squash, wsize, tcp, version.

Have a look at the nfs or the exports man page and the NFS Howto for a more detailed description of what these options mean.

/etc/dfs/dfstab: A typical dfstab file on Solaris should look like the following:

```
share -F nfs -o rw=193.250.160@,ro=193.250.161@ /export/home
share -F nfs -o ro=193.250.160@,root=193.250.161.132 /export/engineering
```

Of course, these options are discussed in detail at the dfstab man page.

The directories we want to export are /usr/local/linux/base/usr, /usr/local/linux/base/opt, /usr/local/linux/base/lib/modules and /home, assuming that you've followed the suggested structure.

Optimising NFS

Of course, this is none of our business but here are some general principles:

- Reduce the TCP window size (parameter wsize for Linux) to whatever is closest to the MTU of your network type. For Ethernet, a good value of wsize is 2048 bytes as long as the MTU is 1536 bytes. This is generally a good idea because the main traffic load between the clients and the server consists of little packets and only in the case of starting large programs like X or StarOffice there is a big number of fragmented packets. Of course this may vary in your case, according to the needs of your users.

- If you plan to have a large installation, break the space for your workstations into 2 or more SCSI disks. This will allow consequent writes and reads on both disks, increasing responce and reducing latency before a request completes.

- Always use NFS v3 over TCP. The main reason for migrating from v2 to v3 is the writeback case it offers on both the workstation and the server. Also, mounting NFS over TCP lets you use the first recommendation.\end{itemize} For further optimising use a packet analyzer like Ethereal or tcldump and decide your needs. By the way, Sun has written an excellent guide to optimizing NFS performance which, although emphasised on Solaris, is applicable to every modern Unix and is accessible online at http://docs.sun.com.

2.2 Setting up the DHCP/BOOTP server

Although there are many DHCP or BOOTP servers 'out there', some of which are proprietary, the best option is to use the reference IETF DHCP server. It is the least vulnerable and the most extensible DHCP available. The main server configuration is done through the /etc/dhcpd.conf file. This file is divided into two sections, the general server configuration and the host specific configuration. A typical dhcpd.conf file looks like this, in case that the DHCP/BOOTP server is used in BOOTP mode:

```
subnet 193.250.160.0 netmask 255.255.255.0 {
    range 193.250.160.10 193.250.160.12;
}
```
This structure is fairly easy to be understood by everyone. For every diskless client we have to supply the program with a 'host' declaration providing a pair of hardware and IP addresses. The host name provided in the 'host' statement can be everything, but there is a conversion to use the real host name of the client having the specific IP. The range statement in the subnet declaration is not necessary to be the range that you want your clients to have. In fact, if these clients are normal workstations with an operating system that during its boot uses DHCP to obtain an IP address it is not recommended to have the same IP for their operation as diskless clients. If you have specific needs, have a look at dhcpd.conf man page.

Another difficulty is how to obtain the IP – MAC address pairs for a large network. The solution is a nice little program called arpwatch. This program runs at the background and keeps track of the IP – MAC address pairs of the computers that your computer has contacted in a file that you have specified. The only thing you have to do is to ping the computers you want. At Appendix B there is a script that starts arpwatch, pings a range of subsequent IPs and creates the dhcpd.conf file. If you want to do it manually, start arpwatch when your network is at its peak of usage and wait for some time. On a shared medium network (Ethernet, Tokenring) arpwatch will track down all different IP ’s and hardware addresses.

2.3 Preparing the base system

To prepare the base system just install your favorite distribution to a mountable partition on a hard disk with a Unix like operating system already installed. Install all the programs you want to be available to your users. Then you have to transfer the whole partition preserving the links and the character or block devices. This is best done using the tar program. Boot the previously installed system and execute the following command, assuming that you have mounted the new partition at /mnt:

    tar cpvf system.tar /mnt/.

This command will create a tar archive at the current directory with the whole system to be served to the diskless clients. Then just copy the tar archive to the server using a CDROM or through the network and extract it at the base directory. The command to do this is:

    tar xvf system.tar /usr/local/linux/base

3. Setting up the clients

3.1 Errata

In order to setup the clients, we have to work on the base system. First, we will make some modifications to the startup scripts by hand and second we will boot a workstation with the base system to make sure it works
and to polish some details. Note that this part is very distribution specific and perhaps some of those described here are not applicable to your case. I can only guarantee that this works for SuSE 7.0. Please, feel free to send me distribution specific copies of this page!

### 3.2 Fiddling with scripts and files!

After `init` is started, it executes a script described in `/etc/inittab`. This script has a very specific job to do: Bring the system in a state that other programs can be started. In most distributions I can think of this script does the following:

1. Mounts the `/proc`, `/dev/pts` and `swap` filesystems.
2. Activates raid arrays and `fscks` the root filesystem.
3. Adjusts the clock.
4. Starts the kernel daemon for autoloading of modules.
5. Executes user defined client scripts.
6. Set some kernel parameters.

On most distributions I have checked this script is very well commented and it is possible for an experienced user to remove some lines that are not wanted or not applicable during a network boot. I've also noticed that all programs started do not require the `/usr` directory to be mounted. If you are trying to netboot a host you must do the following modifications to this script:

- Remove all entries that do `fsck` or initialise raid arrays, and add to the top of the script this command: 
  ```sh
  mount -o remount,rw / because the client has to have rw access to the root directory when it boots.
  ```
- Do not let the kernel daemon start until all partitions are mounted.
- Mount a swap partition. This is described later.
- Start the portmapper. If your system has a specific directory for starting bootup scripts, place the `portmapper` startup script there giving it the highest priority possible, for example: `ln -s /etc/rc.d/portmap /etc/rc.d/boot/S01portmap` if you are using SuSE.
- Place the `NFS` filesystem mounting script in the system specific directory for boot scripts with priority lower than the portmapper, for example: `ln -s /etc/rc.d/nfs /etc/rc.d/boot/S02nfs` for SuSE.
- Remove all entries that automount local partitions, and all entries that start an automounter daemon for RedHat.

#### How to setup a swap partition

This is tricky business! Swapping over NFS is not allowed by the kernel and not functioning either. You cannot use `swapon` on files that are on an NFS mounted filesystem. We have to do some tricks to enable it:

1. Create the swap file. Its size can be variable but for a machine with 128 MB of RAM a swap size of 40–50 MB seems reasonable. The command to create the swap file is: `dd if=/dev/zero of=/var/swap bs=1k count=Xk` where X stands for the number of MB your swap should be.
   It is also a necessity to put the swap file under `/var` as long as it is mounted at boot.
2. Format the swap file using the `mkswapfs` command.
3. Initialise a loopback device using the swap file. The command is `losetup /dev/loop0 /var/swap`.
4. Mount the loopback device with the command `mount /dev/loop0 swap`.

You have to initialise a swap partition at the very beginning of the boot process. So place commands 2–4 somewhere near to the top of the startup script. The first command is very time consuming, especially in the
case of a loaded network so just copy a swap file in the base system and do not delete it when you create directories for each host.

**Modifying /etc/fstab**

The `/etc/fstab` file contains entries for automounting file systems at boot. In our case, we have to place the following lines at the end of it:

```plaintext
server_IP:/usr/local/linux/base/usr /usr nfs nfsvers=3,wsize=2048,tcp 0 0
server_IP:/usr/local/linux/base/opt /opt nfs nfsvers=3,wsize=2048,tcp 0 0
server_IP:/usr/local/linux/base/lib/modules /lib/modules nfs nfsvers=3 wsize=2048,tcp 0 0
fileserver_IP:/home /home nfs nfsvers=3,wsize=2048,tcp 0 0
```

Also, do not forget to comment out lines that mount local partitions. Save this file as `/etc/fstab.new` because it should not be activated yet, as long as we have to boot the base system first.

**Copying password files**

You must provide the system with to files to let the users perform a login. To do this just copy the files `/etc/passwd` and `/etc/shadow` from your file server to the base system. Notice that you have to do it every time you add a user to the system, or a user changes his/her password, so can best be done by creating a cron job.

### 3.3 Booting the base system

To boot the base system we have to create a boot disk first. Go to the next section and create a boot disk as recommended. Please, change the 'append' line to this one:

```plaintext
append init=/sbin/init root=/dev/nfs
   ip=X:Y:195.251.160.254:255.255.255.0::'off'
   nfsroot=Y:/usr/local/linux/base vga=0x318
```

(Of course, in a sigle line)

where X stands for an unused IP address in your network and Y for the IP address of the NFS server. Of course, you have to export the `/usr/local/linux/base` directory from the NFS server with the `rw,no_root_squash` options. Now boot the base system. Everything should work OK, but I don't think that there is a possibility that you succeeded from the first boot! There are many obscure points, that you have forgotten to edit or I have forgotten to mention.

This is the standard method to boot the base system and to add programms or a new kernel to your installation. So backup the files you have edited as well as the boot disk image.

After succeeding to boot the system, you are in a complete linux enviroment. Login as root and enjoy a first ride in your newly created system! Now comes the hard time... You have to disable some services that startup automatically and remove some programms not needed by the users.

### 3.4 Configuring the system

Nearly all distributions start these services:
• **inetd**, the Internet superdeamon responsible for starting other deamons like telnet, ftp etc.
• **syslogd**, the logging deamon. Not needed on a diskless client not needed because all the modifications are done to files easily replacable.
• **httpd**, the apache webserver. Not needed for obvious reasons.
• **dhcpcd**. Needed for automatic acquisition of an IP address. At out case, this is done by the kernel.
• **lpd**, the line printer deamon. This is needed only when you have a printer connected to a host. In most cases, this is not needed.

Also, according to your installation, there may be started sshd, nscd, cupsd and other network services not needed on clients. To disable these services, remove their entries from the runtime directory under /etc/rc.d/X. There is a more elegant way to do this under SuSE or RedHat, using Yast or Linuxconfig. For Yast, go to System administration --> Change configuration file and using search locate the entries for every service you want to stop.

Then, uninstall all these services from the base system. The only service that seems reasonable to me to be left running is the NameServer caching deamon, which is able to reduce network traffic a lot.

Now, you have to edit some files:

- **/etc/resolv.conf** Used to provide a nameserver. Add these entries: nameserver xxx.xxx.xxx.xxx and domain xxxx, replacing x with the correct values.
- **/etc/hosts** Used to match IP addresses to host names localy. Provide the basic servers' names of your network.
- **/etc/nntpserver** Used to provide a news server. Just append the nameserver's hostname.
- **/etc/fstab** Restore the fstab.new file we have created earlier.

### Configuring the language

Perhaps, you do not leave in the US or the UK, like me, so you have to configure the language. This is simply done through the .profile file. Just add the following: `export LANG="X"` where X is your natural language. Then, download a console font which supports your codepage and set, with the help of Yast, the keyboard keymap. Copy .profile to /etc/skel of the file server or to all the users' home directories.

### The X window system

If you want to provide a working X enviroment for clients with different graphics hardware, you have to use the XFBDev server. If you followed the instructions on how to create a boot disk, you would now be in framebufer mode at 1024x768@16M colors, which is sufficient for use with X windows. Now, you have to configure the X server to load the framebuffer driver. SuSE provides an exellent tool for configuring X whether it might be version 3 or 4. It is called sax for X 3.3.x and sax2 for X 4.x. To use XFBDev driver start sax with the `-s XF86_FBDev` option and configure the server according to your hardware. In case you do not use SuSE, most of the work must be done by hand. Create a basic /etc/X11/XF86Config file using xf86config. Please choose entries that are as much as possible closer to your needs. Then edit the /etc/X11/XF86Config. This file is devided into sections that start with the keyword 'Section' and end with 'EndSection'. Do the following modifications:

- **Section "Files"**: Add the path to the direcory where you've put your fonts.
- **Section "Module"**: Load the GLX module if you want REALLY SLOW Open GL graphics (Load glx)!
- **Section "InputDevice, Driver="mouse""**: Add the following lines if you want to use a wheel mouse:
• Section "Device": Replace everything with the following:
  BoardName "AutoDetected"
  Driver "fb"
  Identifier "Device[0]"
  VendorName "AutoDetected"

• Section "Modes": Replace everything with the following:
  Identifier "Modes[0]"
  Modeline "1024x768" 71.39 1024 1040 1216 1 400 768 768 776 802 832

• Section "Screen": Replace everything with the following
  DefaultDepth 16
  SubSection "Display"
    Depth 16
    Modes "1024x768"
  EndSubSection
  Device "Device[0]"
  Identifier "Screen[0]"
  Monitor "Monitor[0]"

• Section "ServerLayout": Replace everything with the following:
  Identifier "Layout[all]"
  InputDevice "Keyboard[0]" "CoreKeyboard"
  InputDevice "Mouse[1]" "CorePointer"
  Screen "Screen[0]"

and then replace the first argument of the InputDevice directives with the identifiers which can be found earlier in the file.

I thing that it should be a working configuration for framebuffer systems. For further reference take a look at the XF86Config and the xf86cfg4 man pages. You will find a working XF86Config file at Appendix C.

**Configuring network access for KDE2**

KDE is the most extensible, configurable and internet enabled window manager available, even if we count some commercial ones that are proud of it! To download KDE, ftp to ftp.kde.org and get the rpms for your distribution. There, you can also find vanilla sources and other related projects.

The main configuration to KDE is done through the K Control Center. There you can find options for configuring the fonts, colors, backgrounds etc. The most important thing you can configure is the LAN browsing deamon that KDE incorporates, lisa. There is also a readme file under $KDE2ROOT/share/apps/lisa. After you configure lisa, you have to make it (or her?) start in the background every time the computer is started. Find the lisa’s configuration file under /root. Copy it under /etc. Afterwards, place the command lisa -c /etc/lisa.conf at the /etc/rc.d/boot.local file, or the similar for your installation. Now tell me, which is easiest to search a network Windows or Linux?

If your users are coming from the Windows world, they are familiar to find programms at the damned 'Start' menu. To make their transition easy, edit the KDE menu with the Menu Editor programm and add or remove applications there. Then, copy the .kde2 directory from you directory to the /etc/skel directory of your file server. Every new account you create will have access to the menu (and the settings) you have created.
4. **Preparing the boot disk**

To prepare a boot disk we just want a kernel, syslinux and a 1.44MB diskette. Syslinux is tiny boot loader, designed specifically to boot a kernel and pass some arguments through its command line using a diskette. As we will see it very easy to configure, too.

### 4.1 Building a kernel

Always choose the newest kernel to build. As of this time of writing (Wed Sep 12 17:28:22 2001) the newest kernel is 2.4.9. Building an older kernel can only save you time updating the necessary programmes. Also, be sure you have the program versions described in `/usr/src/linux/Documentation/Changes`. It is a good idea to compile the kernel using the base system to be served. The kernel can be build according to your needs of drivers, but it must contain the following options:

- **Build in support for the client’s network card** (Network device support ---> Select your card driver).
- **Build in support for the BOOTP protocol** (Networking options ---> IP: kernel level autoconfiguration ---> IP: BOOTP support).
- **Build in support for NFS and root over NFS** (File systems ---> Network File Systems ---> NFS file system support and File systems ---> Network File Systems ---> NFS file system support ---> Root over NFS).
- **Build in support for loopback devices** (Block devices ---> Loopback device support).

Do not forget to compile in the VESA framebuffer driver. Then go on with the familiar kernel compilation routine. Unless you have build the kernel using the base system, copy all the modules created to the base/lib/modules directory of the exported directory structure. The new kernel resides at `/usr/src/linux/arch/i386/boot`.

You also have to set the root device to your kernel. You have to use the `rdev` programm. Execute the following commands:

```bash
mknod /dev/boot255 c 0 255
rdev /path/to/kernel/file /dev/boot255
```

### 4.2 Creating the boot disk

Now, we have to use the syslinux programm. Insert a disk into the first floppy drive and run:

```bash
syslinux -s /dev/fd0
```

Mount the floppy and notice that syslinux has written 2 files: `syslinux.cfg` and `ldlinux.sys`. The second is the boot loader executable. The `syslinux.cfg` is the programm configuration file. A typical structure for that file is the following:

```bash
default linux
    append init=/sbin/init root=/dev/nfs
    nfsroot=195.251.160.10:/usr/local/linux/ws/
    %& vga=0x318
```

```
prompt 1
    timeout 30
```
The default statement is the kernel name to be booted and the append is the command line to be passed to the kernel. Now, you have to copy the kernel you have created to the floppy and rename it to 'linux'.

### 4.3 The kernel command line

To boot a diskless client, its kernel must have the following command line options:

- **init=/sbin/init**: If your init program is elsewhere just change the path.
- **root=/dev/nfs**: An alias to say the kernel that it has to mount its root directory over nfs
- **ip**: This command line option tells the kernel how to get its IP address and which is the NFS server's address
- **nfsroot**: Tells the kernel to mount this directory as its root. The % is an alias to the host's IP address.
- **vga**: If you want to be able to start X windows in framebuffer mode, switch to a framebuffer mode. The one given stands for 1024x768@16M colors.

All these options are discussed in detail in `/usr/src/linux/Documentation/nfsroot.txt`. Read it and adjust the given command line to your needs.

Now you have created the boot disk you are ready to test the system you have build. Start the NFS and BOOTP services and boot a client with the boot disk. No one has been able to do it from the first time. So go on to the next section!

---

### 5. The magic time

In this section will be discussed all the problems that you have and the changes that you propose to the installation. Please feel free to email me and ask about any difficult or not mentioned points in this document. My email is cs98011@icsd.aegean.gr

**Q**: A DHCP is already running. How do I configure BOOTP, so as no interaction is made with the DHCP?

**A**: This was the main problem I faced when I installed the system on a running network. DHCP and BOOTP use the same port. When a windows client boots, it issues a DHCP/BOOTP request to locate its IP (of course in case of dynamic IP). When the DHCP server responds, it also returns the IP's of DNS servers, print servers and Domain Controllers. My BOOTP server was responding faster than the Microsoft DHCP server, and so Windows clients were unable to locate their Domain controller. This resulted to users not being able to login! The solution described here was donated by D. Spinellis.

Open the `/usr/src/linux/net/ipv4` file. This is where all BOOTP autoconfiguration is done. Search for `udph.source, udph.dest` variables. You will see that they are set to the standard 67/68 request/responce ports. Change BOTH values so they use an unused UDP port in your network. A good port pair that no application uses it is 967/968. Now, start your DHCPd with the `-p 967` option. Everything must be working OK!
6. **Other Stuff**

6.1 **Contributors**

- Diomidis Spinellis: Structure and typographical corrections, the DHCP/BOOTP conflict resolution.

6.2 **Copyrights**

This document is GNU copylefted by [Georgios Gousios](mailto:georgios.gousios@utoronto.ca).

It is covered by the GNU documentation licence.

Permission to use, copy, distribute this document for any purpose is hereby granted, provided that the author's / editor's name and this notice appear in all copies and/or supporting documents; and that an unmodified version of this document is made freely available. This document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, either expressed or implied. While every effort has been taken to ensure the accuracy of the information documented herein, the author / editor / maintainer assumes NO RESPONSIBILITY for any errors, or for any damages, direct or consequential, as a result of the use of the information documented herein.

6.3 **Contacting the author**

The author may be contacted via e-mail. For any change, question, error that must be corrected please feel free to contact me. For every contribution you make for this document, your name will be mentioned in the contributors section.

6.4 **Changelog**

- v0.8, Thu May 24 17:37:13 2001: First draft written.
- v1.0, Fri May 25 01:36:25 2001: The first version is complete (in HTML).
- v1.05, Thu Jul 19 19:09:58 2001: Structure and typos corrections. Also, transferred to LaTeX.

7. **Appendix**

7.1 **Appendix A – A script for creating host directories**

```bash
#!/usr/bin/bash
# This is a script for creating host directories using the
directory scheme illustrated before in this document.
# It is written on Solaris and I did not test it on Linux.
# Execute it at the ws directory.
# Needs as input a file containing space separated IP
# addresses named addr, for example bash# ./script addr
# This file must be like this: 195.251.160.10 195.251.160.11 195.251.160.13 ....

echo "Creating the tar archive"; echo
```

6. Other Stuff
cd base
tar cpf linux.tar ./bin ./dev ./etc ./lib ./sbin ./var
mv linux.tar /usr/local/linux/ws/linux.tar
cd ..

echo "Creating host directories"; echo

for addr in $(cat addr)
do
  echo "Working on host $addr"
  mkdir $addr
  cd $addr
  echo "−−−Creating necessary directories"
  mkdir boot
  mkdir cdrom
  mkdir floppy
  mkdir home
  mkdir opt
  mkdir proc
  mkdir root
  mkdir tmp
  mkdir usr
  echo "−−−Extracting tar archive"
  ln −s ../linux.tar ./linux.tar
  tar xf linux.tar
  rm linux.tar
  echo "−−−Removing unnecessary files"
  rm −R ./lib/modules/*
  rm −R ./var/yp
  rm −R ./var/X11R6/sax
  rm −R ./var/tmp
  rm −R ./var/state/dhcp
  rm −R ./var/squid
  rm −R ./var/run/*
  rm −R ./var/opt
  rm −R ./var/named
  rm −R ./var/mysql
  rm −R ./var/lib/amanda
  rm −R ./var/lib/codadmin
  rm −R ./var/lib/firewall
  rm −R ./var/lib/apsfilter
  rm −R ./var/lib/gdm
  rm −R ./var/lib/misc
  rm −R ./var/lib/nobody
  rm −R ./var/lib/pcmcia
  rm −R ./var/lib/pgsql
  rm −R ./var/lib/rpm/*
  rm −R ./var/lib/setup
  rm −R ./var/lib/wvdial
  rm −R ./var/lib/wwwrun
  rm −R ./var/lib/xdm
  rm −R ./var/lib/xkb
  rm −R ./var/lib/YaST/*
  rm −R ./var/lib/zope
  rm −R ./var/log/*
  rm −R ./var/cache/*
  rm −R ./var/games
  rm −R ./var/adm/*
7.2 Appendix B – A script to create the dhcpd.conf file using arpwatch

#!/bin/bash
# A script that starts arpwatch, pings a range of addresses and creates an
# /etc/dhcpd.conf file from the output of arpwatch.
# The arp.dat2dhcpd.conf program is described later.
# Do not forget to edit the i variable and the while statement to specify
# the range of the addresses you want to ping

i=128;

echo "Starting arpwatch";
echo

arpwatch

while [ "$i" -lt 253 ]
  do
    addr=195.251.160.$i
    echo "Now pinging $addr"
    ping -c 5 $addr >/dev/null
    i=$(($i+1))
  done

echo

exit

killproc arpwatch

echo "Creating /etc/dhcpd.conf"

cat /var/lib/arpwatch/arp.dat |arp.dat2dhcpd.conf >/etc/dhcpd.conf

The arp.dat2dhcpd.conf script

#!/usr/bin/perl −n
($ether, $ip,$stup1,$name) = split;
if ($name eq "") {
  print " host host$i {
    hardware ethernet $ether;
    fixed-address $ip;
  }
"
  $i++;}
else{
  print " host $name {
    hardware ethernet $ether;
    fixed-address $ip;
  }
"
}

7.2 Appendix B – A script to create the dhcpd.conf file using arpwatch
# This file should let X 4.0.1 work in 1024x768@16M colors
# with the fbdev driver using the linux's framebuffer

Section "Files"
  RgbPath       "/usr/X11R6/lib/X11/rgb"
  FontPath      "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
  FontPath      "/usr/X11R6/lib/X11/fonts/local"
  FontPath      "/usr/X11R6/lib/X11/fonts/misc:unscaled"
  FontPath      "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
  FontPath      "/usr/X11R6/lib/X11/fonts/Type1"
  FontPath      "/usr/X11R6/lib/X11/fonts/URW"
  FontPath      "/usr/X11R6/lib/X11/fonts/Speedo"
  FontPath      "/usr/X11R6/lib/X11/fonts/misc"
  FontPath      "/usr/X11R6/lib/X11/fonts/75dpi"
  FontPath      "/usr/X11R6/lib/X11/fonts/100dpi"
  FontPath      "/usr/X11R6/lib/X11/fonts/PEX"
  FontPath      "/usr/X11R6/lib/X11/fonts/cyrillic"
  FontPath      "/usr/X11R6/lib/X11/fonts/latin2/misc"
  FontPath      "/usr/X11R6/lib/X11/fonts/latin2/75dpi"
  FontPath      "/usr/X11R6/lib/X11/fonts/latin2/100dpi"
  FontPath      "/usr/X11R6/lib/X11/fonts/latin7/75dpi"
  FontPath      "/usr/X11R6/lib/X11/fonts/kwintv"
  FontPath      "/usr/X11R6/lib/X11/fonts/truetype"
  FontPath      "/usr/X11R6/lib/X11/fonts/uni"
  FontPath      "/usr/X11R6/lib/X11/fonts/ucs/misc"
  FontPath      "/usr/X11R6/lib/X11/fonts/ucs/75dpi"
  FontPath      "/usr/X11R6/lib/X11/fonts/ucs/100dpi"
  FontPath      "/usr/X11R6/lib/X11/fonts/xtest"
EndSection

Section "ServerFlags"
  AllowMouseOpenFail
EndSection

Section "Module"
EndSection

# This section is no longer supported
# See a template below
# Section "XInput"
# EndSection

Section "Keyboard"
  Protocol      "Standard"
  XkbRules      "xfree86"
  XkbModel      "microsoft"
  XkbLayout     "us"
EndSection

Section "Pointer"
  Protocol              "PS/2"
  Device                "/dev/psaux"
  SampleRate            60
  BaudRate              1200
  Buttons               5
EndSection

Section "Monitor"
  Identifier    "Primary−Monitor"
  VendorName    "Unknown"
ModelName "Unknown"
HorizSync 29−64
VertRefresh 47−90
Modeline "1400x1050" 59.93 1400 1416 1704 1816 1050 1050 1055 1097
Modeline "1280x960" 59.90 1280 1296 1552 1664 960 960 965 1003
Modeline "1600x1000" 59.90 1600 1616 1968 2080 1000 1000 1004 1044
Modeline "1024x864" 59.89 1024 1040 1216 1328 864 864 870 902
Modeline "800x600" 58.55 800 816 928 1040 600 600 608 626
Modeline "1152x864" 59.99 1152 1168 1384 1496 864 864 870 902
Modeline "1280x1024" 59.90 1280 1296 1552 1664 1024 1024 1029 1070
Modeline "640x480" 37.44 640 656 720 832 480 480 486 501
Modeline "1024x768" 59.89 1024 1040 1216 1328 768 768 774 802
Modeline "1600x1200" 59.90 1600 1616 1968 2080 1200 1200 1204 1253
EndSection
Section "Device"
  Identifier "Primary−Card"
  VendorName "−−−AUTO DETECTED−−−"
  BoardName "−−−AUTO DETECTED−−−"
EndSection

Section "Screen"
  Driver "fbdev"
  Device "Primary−Card"
  Monitor "Primary−Monitor"
  DefaultColorDepth 16
  SubSection "Display"
    Depth 32
    Modes "default"
  EndSubSection
  SubSection "Display"
    Depth 24
    Modes "default"
  EndSubSection
  SubSection "Display"
    Depth 16
    Modes "default"
    Virtual 1024 768
  EndSubSection
  SubSection "Display"
    Depth 8
    Modes "default"
  EndSubSection
EndSection

Section "Screen"
  Driver "fbdev"
  Device "Primary−Card"
  Monitor "Primary−Monitor"
  DefaultColorDepth 16
  SubSection "Display"
    Depth 32
    Modes "default"
  EndSubSection
  SubSection "Display"
    Depth 24
    Modes "default"
  EndSubSection
  SubSection "Display"
    Depth 16
    Modes "default"
    Virtual 1024 768
EndSection
EndSubSection
SubSection "Display"
  Depth 8
  Modes "default"
EndSubSection
EndSection