The NetWinder Firmware-HOWTO

Ralph Siemsen, ralphs@netwinder.org $Revision: 1.13 $, $Date: 2001/12/11 07:30:11 $

This manual explains how to use and update the NetWinder firmware and how to make use of the initrd capability for disk-less booting.

Contents

1 Introduction

1.1 What the firmware does

The NetWinder contains a 1 MB flash memory chip which holds the program code for initializing the system when first powered on. This 'firmware' is responsible for such activities as turning on the video display and loading the core parts of the operating system from disk into memory. It is roughly equivalent to the BIOS on ordinary PC's.

The firmware on the NetWinder is actually a small Linux kernel, with support for hard disk and network access. The main purpose of this 'minikernel' is to fetch the main kernel and to set up the root filesystem. The minikernel can load these resources from either the hard disk or from the network.

The minikernel also supports some advanced booting options: a small root filesystem can be stored in the unused portion of the flash memory, or it can be fetched from the network and stored into RAM. Seasoned Linux users will recognize this as a slight variation on the standard `initrd` facility.

1.2 Terminology

The terms 'firmware', 'nettrom', and 'flash memory' are used throughout the document and generally refer to the same thing, namely the contents of the flash memory chip. San Mehat wrote the firmware and called it 'NeTTrom' which is where the name originated.

The firmware has a number of user-configurable parameters. These parameters are sometimes called the 'firmware settings' or 'nettrom settings'. Perhaps the notation should be standardized, but the wrong word slips out far too often, so you might as well accept both names.

The terms 'flashing' or 'burning' are used to refer to the process of reprogramming the contents of the flash memory chip. Sometimes it's called 'popping' as well, though I will try to avoid that term.

2 Using the firmware

This chapter describes the firmware settings that are commonly used for everyday operation of the NetWinder. The examples are written for the 2.0.X versions of the NetWinder firmware; older versions such as 1.3pl4 should be updated (consult the 'Updating the firmware' chapter for details). The firmware version number is one of the first things to be displayed on the screen when the NetWinder boots up.

2.1 Accessing the firmware settings

The firmware settings are accessed by interrupting the normal boot process when the message 'Press any key to abort autoboot' is displayed. Pressing the space bar or some other key at this point will cause the firmware control prompt to appear. From this prompt, various commands can be issued to display and change firmware settings.

Alternatively, the same firmware control prompt can be accessed on the serial port. If there is no keyboard plugged into the NetWinder, then the firmware will assume 'headless' operation and will redirect its output to the serial port at 19200 baud, 8 data bits, no parity. (Note: the speed has been increased to 115000 in version 2.0.8h and beyond).

2.2 Active, stored, and default parameters

The `printenv` command can be used to display the parameters and their current values. The listing shows the parameter name in the first column, the active value in the middle column, and the stored value in the right column. The firmware actually maintains three separate sets of values for each parameter: active, stored, and default. The active settings are stored in RAM and apply to the current session only. The active settings can be changed using the `setenv` command.

The active settings can be made permanent with the `save-all` command, which copies the active parameters into the stored ones. It is also possible to retrieve the stored values into the active ones using the `load-all` command. Finally, it is possible to load factory default values into the active parameters using the `load-defaults` command.

2.3 Usage examples

The following examples show some typical configurations for the benefit of those who don't want to read a long description of each setting (for all the gory details, please see the 'Command reference' chapter). All of the examples begin with `load-defaults` to clean the slate, and end with `save-all` to make the settings permanent. Neither of these commands are strictly necessary, and experts may chose to leave them out. I've included them here to ensure that the examples will work, regardless of what state your system happens to be in.

2.4 Factory defaults

For starters, here is how to get the machine back to the factory default settings. This means that the kernel will be read from the file `/boot/vmlinux` on `/dev/hda1` (the first partition on the hard disk), and that `/dev/hda1` will also be mounted as the root filesystem.

```
load-defaults
save-all
boot
```

(Note: the default values are not suitable for machines that have just been upgraded from a pre-2.0 version of the firmware, such as 1.3pl4. See the notes regarding older firmware in the 'Upgrading firmware' section for more information).

2.5 Using a different kernel

Suppose you've compiled a new kernel, called `my_new_kernel` and located in the `/boot` directory. To boot this new kernel, you would use the following NeTTrom commands:

```
load-defaults
setenv kernfile /boot/my_new_kernel
save-all
boot
```

Remember that the `save-all` is optional - if you leave it out, the new kernel will be loaded, but next time you reboot, the old kernel will be loaded. When testing out new kernels for the first time, this is probably a good feature!

2.6 Booting a different partition

Normally, the kernel and root filesystem are read from `/dev/hda1`, but there is no reason why it has to be this way. Suppose that you had downloaded a new disk image (a newer build, perhaps) and you've untarred it to `/dev/hda3`. You can boot the new image as follows:

```
load-defaults
setenv kerndev /dev/hda3
setenv rootdev /dev/hda3
save-all
boot
```

2.7 Booting an NFS disk image

It is possible to boot the NetWinder with the root disk mounted via NFS from another server on your network. This might be useful for recovery purposes, for example. To make this method work, the 'other server' must have an IP address and it must export a suitable filesystem for the NetWinder to boot from. Suppose the server has an IP address of 1.2.3.4, and it is exporting a disk image as `/diskimage`. The following commands will tell the NetWinder firmware to boot from this NFS image:

```
load-defaults
setenv rootconfig nfs
setenv rootpath 1.2.3.4:/diskimage
```

In order to be able to talk on the network, the NetWinder needs to be assigned an IP address (and a netmask). It is possible to use DHCP to assign these addresses, but for simplicity, a static IP will be assumed. Supposing the NetWinder has an IP address of 5.6.7.8 with a netmask of 255.255.0.0, then the NeTTrom commands would be:

```
setenv netconfig_eth0 flash
setenv eth0_ip 5.6.7.8/16
save-all
boot
```

Note that the netmask is expressed in IPv6 style - 255.0.0.0 becomes /8, whereas 255.255.255.0 becomes /24.

This example still loads the kernel from the local hard disk, but then boots off the network via NFS. The hard disk isn't used after the kernel has been fetched.

2.8 Kernel fetched via TFTP

The firmware can load its kernel from the network (as opposed to from the local hard disk). To make this work, the NetWinder needs an IP address and a netmask, and there needs to be a suitable boot server available on the network. The boot server must be able to transfer a kernel via the TFTP protocol. On the NetWinder, this boot option is enabled with the commands:

```
load-defaults
setenv kernconfig tftp
setenv kerntftpserver 10.11.12.13
setenv kerntftpfile /tftpboot/vmlinux-netwinder
```

This assumes that the boot server's IP address is 10.11.12.13 and that the filename of the kernel on the bootserver is `/tftpboot/vmlinux-netwinder`. As shown in the previous example, the NetWinder must be assigned an IP address and a netmask so that it can communicate over the network.

```
setenv netconfig_eth0 flash
setenv eth0_ip 5.6.7.8/16
save-all
boot
```

This sequence will fetch the kernel from the server and store it in RAM. The system will then boot and mount the local hard disk `/dev/hda1` as the root device.

2.9 Complete diskless booting

The NetWinder can be booted without using the hard disk at all (or even without a hard disk installed) by combining the two previous examples. This is useful in a number of circumstances, including when you've totally trashed your hard disk :)

As in the previous examples, the IP addresses of the NFS server, NetWinder, and TFTP server will be assumed to be 1.2.3.4, 5.6.7.8, and 10.11.12.13 respectively. Quite frequently, the NFS server and TFTP server will actually be the same machine, so they would have the same IP address. For the sake of clarity, however, separate IPs are shown here.

```
load-defaults
setenv netconfig_eth0 flash
setenv eth0_ip 5.6.7.8/16

setenv kernconfig tftp
setenv kerntftpserver 10.11.12.13
setenv kerntftpfile /tftpboot/vmlinux-netwinder

setenv rootconfig nfs
setenv rootpath 1.2.3.4:/diskimage

save-all
boot
```

The first section configures the NetWinder's network interface (address and netmask), the second block arranges for the kernel to be fetched via TFTP, and the third section sets up the NFS root filesystem.

3 Upgrading the firmware

This chapter explains how to upgrade the firmware to a newer version. To determine which version of the firmware you have, watch the screen when you power up your machine. There are three major releases of firmware currently 'out there': the present series (2.0.x), the old stuff (1.3), and the really old stuff (less than 1.3). Anything pre-2.0 should be updated immediately. Depending on the existing firmware version number, the process for updating will vary slightly: consult the appropriate section below.

3.1 Obligatory warning

Please note that reprogramming the firmware is an inherently dangerous activity that could potentially render your NetWinder completely inoperative. Without the firmware, a NetWinder will not be able to boot itself. Before attempting to reprogram the firmware, back up all important files and prepare yourself for the possibility of having to return your machine for repair. Needless to say, firmware upgrades are done entirely at your own risk!

In practice the flash updating process is really quite safe. Follow the directions given here and things should work out fine. If this is your first attempt to update the firmware, please *read this chapter completely* before you begin. If you still have problems, you can ask for help on the mailing lists or newsgroups (<news://news.netwinder.org>). Statistically speaking, the likelyhood of toasting your machine is less than 1%, based on the number of units returned with dead flashes in the last year.

3.2 Obtaining new firmware

New versions of the firmware can be obtained anonymously from <ftp://ftp.netwinder.org/pub/netwinder/firmware/> and have filenames of the form `nettrom-X.Y.Z.bin` where `X.Y.Z` is the version number. Be sure to use binary mode when downloading the file. At the time of this writing, the current stable nettrom version is 2.1.24. More recent *development* versions can also be found elsewhere on the ftp site (hint: look in Rod Stewart's or Andrew Mileski's directory), but beware that these may not have been extensively tested - and you probably don't own JTAG equipment (just thought I'd remind you).

Some of the `nettrom` binaries are available with an attached 'rescue' filesystem. Especially for those people with small hard disks in their NetWinders, this rescue filesystem is helpful for restoring or updating the disk image. Therefore it is recommended that you use a `nettrom+rescue` image when one is available. The only times you'd really want to use a non-rescue image is when you plan to create your own flashroot filesystem (see the 'Advanced booting' chapter for details).

3.2.1 Checksum verification

You may wish to use the `md5sum` command to generate a checksum of the `nettrom` image you've downloaded to ensure there were no errors in the process. This is important because the flash writing program has no way of knowing if the `nettrom` binary is valid or not - it simply copies the file into the flash memory. To compute the checksum, issue the command

```
md5sum nettrom-2.0.X.bin
```

where `nettrom-2.0.X.bin` should be replaced with the name of the file you've downloaded. Very old NetWinders may not have the `md5sum` program, in which case you cross your fingers and hope for the best. Otherwise, the computed checksum should be compared against those listed in

<ftp://ftp.netwinder.org/pub/netwinder/firmware/md5sums> . If the checksums don't match exactly, delete the `nettrom.bin` and transfer it again. Check that your FTP client is doing a binary transfer, not an ASCII or text transfer.

3.3 Writing the flash

The next step is to transfer the `nettrom.bin` to the actual flash memory chip. This is done using a program called `flashwrite`, which in turn depends on a kernel module called `nwflash.o` (on older systems, it was called `flash.o`). The exact process varies slightly depending on the age of the software on your machine; consult the appropriate sub-section.

You must be logged in as `root` to reprogram the flash, and should not be running any unecessary programs at the time. The flash writing process is very sensitive to timing and might fail if an interrupt occurs at the wrong moment. Before you begin, it is suggested that you unplug any network cables or peripherals, and terminate any active tasks. In other words, the machine should be loaded as lightly as possible.

Once the flash writing process has been started, it should not be interrupted. Should an error be detected, it is important that you *do not reboot* the machine under any circumstances, since the contents of the flash memory are uncertain. Repeat the flash writing process until it succeeds. Consult the troubleshooting section, below, if any errors are encountered.

3.3.1 Recent machines

Recent disk images already contain all the tools necessary to update the flash memory. Simply download a `nettrom` binary from the ftp site and then run the following commands to update the flash memory:

```
insmod nwflash
flashwrite -base64 nettrom-2.0.X.bin 0
```

Replace `nettrom-2.0.X.bin` with the name of the nettrom file you downloaded. By convention, the `nettrom` binary is normally stored in the `/boot` directory but you can put it wherever you wish - you'll have to specify the path accordingly in the `flashwrite` command. Remember that once the flashwrite has started, it must complete successfully before you reboot the machine.

Also note that the final character in the `flashwrite` command line is a zero, not the letter 'o'. This parameter specifies the offset from the beginning of the flash memory - in this case, the data is being written to the beginning of the flash.

3.3.2 Older machines

The process for older machines is pretty similar to that of recent machines (see preceeding section), except the `nwflash` driver is called `flash` and it uses different major/minor numbers. The commands to be typed are therefore:

```
insmod flash
flashwrite -base64 nettrom-2.0.X.bin 0
```

The older flash driver is more susceptible to failure, so be sure to repeat the `flashwrite` command if it fails to write the whole nettrom image. Consult the troubleshooting section below if you have any problems.

3.3.3 Really old machines

On really old machines, the `flashwrite` program and the `flash` driver are not included, since they were changing so often at that time. In this case, it is recommened that you download `nettrom-1.3pl4.tar.gz` in addition to the `nettrom-2.0.X.bin` file. The `tar.gz` file contains all the necessary programs for writing the flash, along with the 1.3pl4 version of `nettrom`. This old `nettrom` should be deleted and the 2.0.X version used in its place:

```
tar zxvf nettrom-1.3pl4.tar.gz
cd nettrom-1.3pl4
rm nettrom
cp ../nettrom-2.0.X.bin nettrom.bin
```

Now the flash can be written in much the same manner as before. The only difference really is that the commands are not on the normal search path, so they must be preceded with '`./`' so they can be found:

```
./insmod flash.o
./flashwrite -base64 nettrom.bin 0
```

As in the case for 'older machines', this version of flashwrite is not very rubust and several tries may be necessary until the entire flash image is successfully written.

3.4 Troubleshooting

Problems during the flash writing process can be attributed to either the `insmod` or the `flashwrite` commands. Obviously, a failure in the former command will also lead to failure of the latter. If the error message doesn't make it clear, you can use the `lsmod` command to verify what modules are loaded. If the `nwflash` or `flash` device is not listed, then the `insmod` failed. Otherwise, check the `flashwrite` command.

3.4.1 insmod problems

Problems with `insmod` fall in two categories: either the executable cannot be found, or the `flash` module can't be found. In the former case, try including the full path, ie. `/sbin/insmod`. Otherwise, the problem is with the `flash` module. Try both `flash` and `nwflash` as the module name. Failing that, try looking for the flash module with a command like `find /lib/modules -name '*flash.o'` and then use the resulting full path name as the module name.

It's also possible to get an error from `insmod` about a kernel version mismatch. This means that the specified `flash` module is not compatible with the kernel. Check to see if any other `flash` modules are available, using the `find` command described above. If not, you can try using the `-f` flag to `insmod`, to force loading of the module. If all else fails, download the `nettrom-1.3pl4.tar.gz` file and consult the instructions for 'Really old machines' above.

3.4.2 flashwrite problems

The most common problem with `flashwrite` is typing in the command incorrectly, or not specifying the name of the `nettrom.bin` file correctly. If the `nettrom.bin` file is not in the current directory, then the relative or absolute pathname must be specified. And following the filename, there must be a zero (as shown in the previous section's examples).

An error message along the lines of 'Can't open /dev/flash' indicates that either the `flash` module wasn't loaded correctly (you can check this with the `lsmod` command), or there is a problem with the device entries. The proper value for the device entries depends on which module you are using: for `flash` the major/minor numbers are 101 and 0, whereas for `nwflash` the numbers are 10 and 160. To set the device nodes up, use the following commands, substituting the appropriate numbers for `MAJOR` and `MINOR`, respectively.

```
cd /dev
rm *flash
mknod -m644 flash c MAJOR MINOR
ln -s flash nwflash
```

Any other errors from `flashwrite` indicate a real problem writing to the flash memory. If this condition persists, try loading the flash driver with debugging turned on: first do `rmmod flash`, then repeat the `insmod` command with `flashdebug=1` suffixed on.

3.5 After writing the flash

Once the `flashwrite` program completes, the update is pretty much done. Ensure that the number of bytes written matches the size of the `nettrom.bin` file you were writing. If the numbers do not match, repeat the flashwrite process again. Otherwise, it is safe to shut down the machine (using CTRL-ALT-DEL for example) and upon reboot the new firmware will be active.

If you are paranoid, you can perform one more test before you reboot. The data can be read back from the flash memory and be compared with the original `nettrom.bin` file. It's pretty unlikely for any discrepancies to turn up, since `flashwrite` already performs such checking, but it doesn't hurt either. You need to know the size in bytes of the `nettrom.bin` file you just flashed.

```
dd if=/dev/nwflash of=actual.nettrom bs=1 count=BYTESIZE
cmp nettrom-2.0.X.bin actual.nettrom
```

Replace `/dev/nwflash` with `/dev/flash` if you've got an 'older' system, and substitute the actual file size for `BYTESIZE`. There should be no output from the `cmp` command; if there is, go back and repeat the entire flash writing process. Keep in mind that this test only verifies that the file was written correctly to the flash, but it cannot protect against having an invalid file in the first place.

3.6 The first reboot

Now it's time to perform ancient tribal rituals (or just cross your fingers) and reboot your machine. With luck, you'll see the white banner screen with the new firmware version number displayed. Shortly thereafter you should see the 'Press any key to abort autoboot' message. Getting that far indicates that your flash reprogramming was successful.

The first time you boot a new version of firmware, it is a good idea to issue the commands `load-defaults` followed by `save-all`. If you don't do this, your old settings will be preserved, but you may see a warning message whenever you enter the firmware menu, and it is possible for there to be some odd side-effects.

3.6.1 Upgrading from pre-2.0 firmware

If you've just upgraded from pre-2.0 firmware, then some special considerations apply. The default parameters in 2.0 firmware assume that the disk is partitioned with `/dev/hda1` as the root filesystem, and that the kernel is also contained on that partition. However, the 1.3 firmware used a different setup - `/dev/hda1` was a dedicated kernel partition, and the root filesystem resided on `/dev/hda2`. It is recommeded to update to the new layout, but that is outside the scope of this document. In order to boot the old-style layout with 2.0 firmware, the following command sequence should be issued the first time you boot up:

```
load-defaults
setenv kernconfig partition
setenv rootdev /dev/hda2
save-all
boot
```

It is recommended that you stop using this partition-based boot method, since it is difficult to recover from a bad kernel. If you don't want to repartition and re-install your whole system, you can still switch to the new boot method. Simply install a NetWinder kernel into the `/boot` directory on your system and instruct the firmware to boot it:

```
load-defaults
setenv kernconfig fs
setenv kerndev /dev/hda2
setenv kernfile /boot/vmlinux
setenv rootdev /dev/hda2
save-all
boot
```

With this setup, you can install many different kernels in `/boot` and switch between them by changing the `kernfile` parameter.

3.7 If it doesn't work...

... Don't panic! In some cases you may still be able to rescue the machine. If the first 32 kB of flash are still intact, then it is possible to serially download a kernel and boot it. Then the flash memory can be reprogrammed again, hopefully with a working image this time!

A second computer is required, along with a null-modem cable. Such a cable can be had at most computer stores, or you can make your own (see the Serial-HOWTO for details). The cable should connect between the serial port on the NetWinder and the serial port on the rescue system.

Next, a terminal program should be launched on the rescue system. It should be configured for 19200 bps, 8 data bits, no parity, and 1 stop bit (For firmware 2.0.8h and beyond, the speed is 115200 bps). Be sure to

turn off any handshaking, both hardware and software. Then turn on or reboot the NetWinder - a number of diagnostic messages should be printed on the serial terminal. If you see nothing, check the cables and the COM port settings. If you really don't get any output, then you flash is truly wrecked, and you'll have to return your machine for repair. Sorry.

Otherwise, you should see the message 'Press * TWICE to abort autoboot' or similar. Some older versions of the firmware used ALT-D instead of an asterisk character. Press the appropriate key, and a Nettrom control menu will appear. You can press '?' for a brief listing of available commands.

The next step is to obtain a NetWinder kernel and put it on the rescue system. If you have older firmware and a 2.0 kernel, please skip down to the next sub-section for further instructions. The "z" option described below is also available in newer firmware versions, but the new "x" option is considerably nicer.

```
x c000
```

The firmware is now expecting a kernel to be transmitted in Xmodem format. Use your terminal program to transmit the file (in minicom the command is ALT-S). You may have to repeat the `x` command since it times out after a short period.

The kernel should download in about 2 minutes at 115200 bps. Once it is done, use the command `j c000` to start the kernel. If you wish to change the root device from the default of `/dev/hda1` (major 3, minor 1), it must be done before the "j" command. Use `d 100` and note that location 110 contains 0301. Change this to the value you want using the `e 110` command.

3.7.1 Older firmware and 2.0 kernels

The official 2.0.x kernels from the ftp site cannot be used directly because they are in ELF format - but the firmware only knows how to deal with an a.out kernel. The `vmelf` utility program (<ftp://ftp.netwinder.org/pub/netwinder/kernel/misc/vmelf.c>) can be used to convert an ELF kernel into a.out form (well, close enough anyways). Just type `make vmelf` to build an executable from the `vmelf.c` source code. It's a hack.

Record the size of the kernel file - it will typically be somewhere between 870000 and 1250000 bytes. Convert the size from decimal into hexadecimal (sorry, that's all the nettrom menu knows about) and write it down. Now, at the Nettrom prompt, use the 'z' command to load the kernel. The arguments are two hexadecimal numbers - the first is always `c000`, and the second should be the size in hex of the kernel file. Don't include the '0x' or '$' symbols in the size value.

```
z c000 HEXSIZE
```

Use the 'upload' feature of your terminal program to actually transmit the kernel file. Notice that the file must be transmitted 'plain binary', meaning that there is no protocol (like xmodem, zmodem, or kermit). It just a raw stream of bytes. This also means there is no error checking. Note that as of firmware 2.0.8, Xmodem is supported; see above for details.

The transmission will take about 10 minutes at 19200 baud. A series of asterisks will be printed as the transfer proceeds. (Yes, there is an option to change the baud rate, but it doesn't work as you would expect. It's best just to stick with the standard rate and wait. After all you should only ever need to do this once). Upon completion of the download, the Nettrom prompt should re-appear. If there is no prompt, or there are lots of extra characters after the Nettrom prompt, then something got out-of-sync during the download.

In this case, you'll probably have to repeat the whole process. But it doesn't hurt to try to boot the kernel anyhow - you might get lucky. The following command will boot the kernel:

```
j c000
```

Hopefully your machine should boot up fine now, and you can go and reprogram the firmware properly. There is no way to pass a command line to the kernel, so the compiled-in default values will be used. Usually the defaults are 16 megs of RAM and a root device of `/dev/hda1` - this should boot most machines. If your setup differs, you can hack different values into the kernel file before you send it down the serial cable. Kernels as of 990121 use a `param_struct` (defined in `include/asm-arm/setup.h`) for passing in the various boot-time parameters. It is possible to modify this structure using the Nettrom debugger commands (type '`?`' for help at the nettrom prompt). The structure is stored at memory location 0x100.

4 Firmware command reference

In this section, the NeTTrom parameters will be explained in detail. It is meant more as a reference guide; for most people, the examples in the 'Using the firmware' chapter should be sufficient. The parameters are grouped into several sections that are logically connected - the same way they are listed when the `printenv` command is used.

4.1 Eth0 configuration

Eth0 is the 10-base-T network interface on the back of the NetWinder. If the minikernel is to do any networking (such as fetching a kernel, or booting from an NFS server), then the interface must be assigned an IP address and netmask in this section. Either static addresses or dynamic (DHCP) can be used.

The first parameter, `netconfig_eth0`, determines how the interface is configured. The default setting of `disk` means that the interface is not configured (inactive). A setting of `flash` means that the address and netmask are specified in the `eth0_ip` parameter. A setting of `dhcp` indicates that DHCP is to be used to configure the interface.

...dhcp parameters should be described here...

The `eth0_ip` parameter contains the network address and netmask value for the eth0 interface. The two values should be separated by a slash, and the netmask should be expressed as a single number (IPv6 style). This field is ignored unless `netconfig_eth0` is set to `flash`.

4.2 Eth1 configuration

Eth1 is the 10/100-base-T network interface. It has a set of parameters that function identically to those of the eth0 interface. Consult the previous section for a full description.

4.3 Routing configuration

These options allow the firmware to contact boot servers (TFTP and NFS) that are not on the same subnet as the NetWinder. Issue the command `setenv route1` with no additional arguments for some examples of how to set it up. This feature has not been widely tested.

4.4 Initial ram disk configuration

The bootloader can load an initial ram disk out of the flash memory. One common use for this feature is to provide a 'rescue' filesystem that can be used to restore the hard disk, without needing to set up an NFS server and a TFTP server.

The `initrd` parameter can be set to either `inactive` or `flash`. In the latter case, the bootloader searches the flash memory for a compressed ram disk and arranges for it to be booted as the root device.

The other initrd options are not implemented as of this writing.

4.5 Kernel configuration

This section determines how the NetWinder will fetch its kernel. The first parameter, `kernconfig`, determines which method will be used to fetch the kernel. It can be set to one of `partition`, `fs`, or `tftp`.

Normally, `kernconfig` is set to `fs`, which stands for 'filesystem'. In this case, the values of the parameters `kerndev` and `kernfile` determine the device and name of the kernel file. The bootloader will look on the specified device and try to load and execute the specified filename. The file should be a valid linux kernel.

Prior to the 2.0 firmware series, the kernel was stored in raw form on a dedicated partition (ie. without a filesystem). Support for this 'legacy' method is available by setting `kernconfig` to `partition`. In this case, the kernel is loaded directly from the device specified by `kerndev`. There is no filename. Obviously, the root device must be on a different device in this case. Most people won't want to use this option.

The third option is to fetch a kernel via TFTP from a server on the network. Setting `kernconfig` to `tftp` enables this option, which also requires that a network interface be configured (see the section above). The ip address of the TFTP server should be stored into the `kerntftpserver` parameter, and the filename (on the server) should be stored in `kerntftpfile`.

The multiple file name fields are provided to make it easy to switch between network and local booting. Once configured, only the `kernconfig` parameter needs to be changed.

4.6 Root device configuration

The `rootconfig` parameter specifies how the NetWinder will obtain its root filesystem. The possible values are either `disk` for local booting, and `nfs` for network booting.

When the `rootconfig` parameter is set to `disk`, then the boot device specified by the `rootdev` parameter will be used. Typically the root device would be `/dev/hda1` or some other hard drive partition. However, any devices that were detected at boot-up may be legitimately specified. For example, a ZIP drive attached on the parallel port could be used as the boot device.

Network booting is enabled by setting `rootconfig` to the value `nfs`. The IP address and the export name for the NFS server should be specified in the `rootpath` parameter (a typical example would be `10.1.2.3:/export/netwinder`).

4.7 Miscellaneous configuration

The `cmdappend` parameter can be used to specify additional options to be passed to the kernel. The contents of this field will be appended to the kernel command line, without any checking done. One common use is

to pass special arguments to the `init` process.

The `passwd` parameter can be used to password-protect the firmware settings, to prevent unauthorized haxors (or young children) from messing with your configuration settings.

5 Advanced booting with `initrd`

This chapter covers the more esoteric booting options, namely those that involve an initial ram disk (`initrd`). The casual user can safely skip this chapter. The features described in this chapter are subject to change.

5.1 Preparing the ram disk

To get started, an empty ram disk should be created. The following example creates an empty filesystem of about 4 MB. There are no particular restrictions on the size of the disk at this point, but making it needlessly large will just consume RAM when the machine is running later on.

```
dd if=/dev/zero of=myramdisk bs=1k count=4096
mke2fs -F myramdisk
insmod loop
mount -oloop myramdisk /mnt
```

The disk can now be populated by copying files into `/mnt`. Keep in mind that for each binary, all necessary libraries and configuration files must be installed as well. Finally, you should create a file called `/mnt/linuxrc` and make it executable. This script will be run when the disk is booted. The `chroot` command is helpful for testing out the disk before installing it. Once you are satisfied, the next step is to unmount, then compress the disk image:

```
umount /mnt
gzip myramdisk
```

This will produce a file called `myramdisk.gz`. There are size restrictions on this compressed ram disk image, as explained in the next section. If your image is too large, you'll have reduce the number of files you put into the image.

5.2 Installation and booting

The compressed ram disk image can either be burned into the flash memory chip, or it can be fetched via TFTP when the NetWinder boots. The first option allows for diskless, networkless booting of a NetWinder, while the second option allows a somewhat larger ram disk to be loaded.

5.2.1 Adding the image to the flash

The compressed disk image can be written to the unused portion of the flash memory. The proceedure is to simply join the compressed image on to the end of a standard `nettrom` binary image, and then to write the combined image into the flash memory:

```
cat nettrom-2.0.X.bin myramdisk.gz >combined.img
insmod nwflash
flashwrite -base64 combined.img 0
```

NetWinders normally have 1 MB of flash memory, with the top 64 kB reserved for configuration data. The total size of '`combined.img`' must not exceed 983040 bytes (that's 1 MB minus 64 kB), otherwise it won't work (and if you were expecting a little warning message to be printed if you exceed the size, guess again...)

To try out the image, reboot the NetWinder and go into the firmware control menu. Issue the command `setenv initrd flash`, save the parameters if you wish, and boot it. If you've done things correctly, your ram disk will be loaded and the `linuxrc` script will be run.

5.2.2 Loading the image by tftp

A somewhat larger initial ram disk can be loaded via TFTP protocol over the network. This option is only available with version 2.0.7 of the NetWinder firmware. The compressed ram disk should be concatenated onto a normal NetWinder kernel. The maximum allowable size of the combined kernel and compressed filesystem is 4 MB (ie. 4194304 bytes). Please note that at this time, only ELF kernels (ie. most 2.0.35 kernels) will work - the 2.2 kernel series are not recognized as ELF and the ramdisk won't be detected. This will be fixed in an upcoming nettrom.

```
cat /boot/vmlinux myramdisk.gz >vmlinux+ramdisk
```

The resulting file `vmlinux+ramdisk` should be transfered to the TFTP server machine, perhaps via ftp or nfs. The NetWinder should then be rebooted and the firmware settings for TFTP booting should be activated:

```
setenv kernconfig tftp
setenv kerntftpserver 10.2.3.4
setenv kerntftpfile vmlinux+ramdisk
```

Of course the IP address and filename will need to be adjusted for your particular setup. For more details, consult the TFTP example in the 'Using the firmware' chapter.

5.3 Serial booting

It is possible to download a kernel via the serial port. This is intended for emergency situations only where the flash memory doesn't contain a valid boot image. See the last section in the 'Updating the firmware' chapter for details.

6 Misc

6.1 Author

The author and maintainer of the NetWinder Firmware-HOWTO is Ralph Siemsen (ralphs@netwinder.org). Please send me any comments, additions, corrections so that the can be included in the next release. The latest version of this document can be obtained from <http://www.netwinder.org/~ralphs/howto/Firmware-HOWTO.html> .

6.2 To-do

The descriptions of some of the boot options in the 'Reference' chapter are lacking. These are the options that I don't normally use, and so I don't have much to say about. To be fixed.

The 'sgml2info' version of this document doesn't show the examples properly - for some reason the linefeeds are removed. Why is this and how do I fix it?

6.3 History

April 20, 1999 (version 1.5): First public release of this document.

April 25, 1999 (version 1.6): Suggested by Woody: Added section on what to do if serial downloading doesn't complete, added note about param_struct at 0x100, and suggested installing new kernel if upgrading from pre-2.0 firmware.

April 26, 1999 (version 1.7): In tftp+initrd section, mention that ELF kernel must be used (and 2.2 currenty doesn't work). Thanks to Jim Studt for bringing this to my attention.

Sep 3, 1999 (version 1.8): Fixed URL's on the ftp site from `pub/ccc` to the new `pub/netwinder`. Added xmodem info in the serial kernel recovery section.

Dec 3, 2001 (version 1.9): Corrected error in `ln -s flash nwflash` example in section 3.4 (thanks to Ron Golan for catching this).

6.4 Contributors

San Mehat (nettwerk@netwinder.org) wrote the original firmware and wrote the preliminary documentation for it.

Woody Suwalski (woody@netwinder.org) maintained the firmware up until version 2.3, and reviewed this documentation.

Andrew Mileski (andrewm@netwinder.org) added in `initrd` support for tftp'ed kernels in version 2.0.7 of the firmware.

6.5 Legal stuff