The NetWinder Kernel-HOWTO

Ralph Siemsen, ralphs@netwinder.org $Revision: 1.7 $, $Date: 2000/11/19 17:14:47 $

This manual explains how to build and install Linux kernels on a NetWinder.

Contents

1 Introduction

This manual describes how compile and install kernels on a NetWinder. The first part explains how to install pre-built binaries, while the second part explains how to compile the kernel from source code. This document focuses on the stable kernels (2.0.x and 2.2.x), although the information applies just as well to the experimental (2.3.x) kernel series.

Previous versions of this document had separate sections for the 2.0 and 2.2 kernel series. These have now been merged into one, with differences noted as appropriate.

There exists another 'Kernel-HOWTO' that explains how kernels are built for Linux in general. It makes good background reading for those people who are unfamiliar with the linux kernel, especially the first two chapters. It can be found in various places on the net, for example `<http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html>`.

2 Binary install / upgrade

This chapter explains where to get and how to install (pre-built) binary kernels on the NetWinder. This is the most common form of upgrade that may be performed, and should be used by most people.

I'd like to stress the previous point. The official NetWinder kernels already contain pretty much all the drivers and functionality that you need (either built-in, or as loadable modules). In general, there should be no need to rebuild the kernel since all NetWinder models have the same hardware in them. If you are a driver author, please consider submitting your driver so it can be included in the official builds.

2.1 Preparation

Before installing a new kernel, it is wise to update the firmware as well. In particular, if you currently have a 2.0 kernel, then the firmware on your machine will probably not be able to boot a 2.2 kernel successfully. The version of the firmware is displayed on initial power-up of the NetWinder.

Firmware versions before 2.0.8h should definately be updated. At the time of writing the current version is 2.1.24, though there have not been any major changes since version 2.1.16. For information on how to update the firmware, please see the `<http://www.netwinder.org/~ralphs/howto/Firmware-HOWTO.html>`.

2.2 Where to get binaries

Official binaries for the NetWinder are available from `<ftp://ftp.netwinder.org/pub/netwinder/kernel/>`. They are distributed in RPM format for ease of installation. Kernels are identified by their official version number (eg. 2.2.13) suffixed with the release date (in the form `YYYYMMDD`). For each release there are two binary RPMS that should be downloaded:

```
kernel-2.x.y-YYYYMMDD.armv4l.rpm
kernel-headers-2.x.y-YYYYMMDD.armv4l.rpm
```

The first package contains the kernel itself and the loadable modules, the second package contains the corresponding header files. The latter will be required to compile applications and hence it is recommended that you always update both of these packages together.

2.3 Installation

Once the two RPM packages have been downloaded to the NetWinder, installation is accomplished with the usual RPM command sequence. Note that you must be logged in as `root` to execute these commands.

```
rpm -ivh kernel-2.x.y-YYYYMMDD.armv4l.rpm
rpm -ivh kernel-headers-2.x.y-YYYYMMDD.armv4l.rpm
```

Note to RPM veterans: unlike most other packages, you *should not* use the `-U` (upgrade) option on the kernel packages, just to be on the safe side. This is because 'upgrade' will remove the old kernel version, possibly leaving you with an unbootable system. See the 'Behind the scenes' section below for more details.

If the RPM fails to install, check that it was downloaded in binary mode - repeat if necessary. If that still doesn't work, then chances are, your machine has an old version of `rpm`. Consider upgrading to a more recent disk image (consult the Disk-Update-HOWTO).

2.4 Trying it out

The NetWinder needs to be rebooted in order to boot the new kernel. However before doing that, it is a good idea to record the names and version of the exising kernel on your system. In case the new kernel doesn't work on your machine, you will then be able to go back to the old version. The following command will list the names of the kernels installed on your system (including the new kernel you just installed)

```
ls /boot/vmlinux-*
```

Write down the names, then reboot your NetWinder and the new kernel should be loaded. It should without any problems. Note that any errors reported after phrase `INIT: entering runlevel N` are problems with user-space applications or configuration settings, and not kernel problems. An error like `VFS: unable to mount root fs` is a kernel problem.

2.4.1 Going back to an old version

If you need to get back to the old kernel, you can do so by interrupting the firmware at the `Autoboot` prompt. The following command should then by entered to boot the old kernel, whose exact name you must specify (you did write it down, right?)

```
setenv kernfile /boot/vmlinux-2.x.y-YYYYMMDD
boot
```

To make this setting permanent you need to modify the symbolic link `/boot/vmlinux` so it points at the name of the kernel you wish to boot. Or you can set the firmware permanently to the specific name (but this is not recommended).

2.5 Cleaning up

While not strictly necessary, the purists among you will undoubtedly want to get rid of the old kernel that isn't needed anymore (assuming the new one works well of course!). This task can be accomplished with the following RPM commands.

```
rpm -e kernel-2.x.y-YYYYMMDD
rpm -e kernel-headers-2.x.y-YYYMMDD
```

Of course you should specify the *old* kernel version and date in the above command. If it fails with an error saying the package is not installed, then most likely the original kernel was not installed via RPM. You may manually delete the appropriate files in `/boot` and `/lib/modules` if you wish, though I'd recommend to just not worry about it - it will not cause any harm to leave them around.

That's all there is to it, you are done!

2.6 Behind the scenes

If you are uncomfortable with RPM doing all the magic work, or just are curious how it really works, here is a brief explanation. The firmware expects to load a kernel called `/boot/vmlinux` from the first partition on the disk (unless you tell it otherwise). Normally `/boot/vmlinux` is a symbolic link pointing to a real kernel (with version number and date).

For each kernel version there is a corresponding set of loadable modules (device drivers). They are located in `/lib/modules/2.x.y-YYYYMMDD`, and the kernel is smart enough to know to look in the right directory (since the kernel knows what its version number is).

When you install a binary kernel RPM, the real kernel and its modules are placed in the appropriate place. Then (in the rpm post-install script) the `/boot/vmlinux` symbolic link is adjusted to point at the newly installed kernel. Upon reboot then new kernel is then loaded.

The kernel-headers package operates similarly. It installs the header files into `/usr/src/linux-2.x.y-YYYYMMDD` and then adjusts the symlink `/usr/src/linux` to point at the specific version.

What about package removal? The RPM program removes all the files that were in the package, but not the symbolic links, since they were created by the post-install scripts. So you can safely uninstall old kernel packages without any side effects, so long as the symlinks still point to real files that are still installed.

3 Building your own

This chapter describes how a kernel is built from source on the NetWinder. The process is almost the same as the usual Linux kernel, with two notable exceptions. (For the impatient, the two differences are: the default configuration file must be copied from `arch/arm/def-configs/netwinder` to `.config` before you begin, and to build the kernel you must use `make Image` or `zImage` and *not* `bzImage`).

3.1 ARM-linux who's who

The mainsteam linux kernel includes some support for the ARM architecture (which the NetWinder belongs to), but it is far from complete. Russell King is the primary developer of the ARM-linux port (see `<http://www.arm.linux.org.uk>`) which includes many other ARM-based systems. Russell distributes patches that can be applied to the mainstream linux kernel, and these can be used as-is on the NetWinder.

The NetWinder-specific port was done by Pat Beirne, San Mehat, Woody Suwalski, and is being maintained now by Phil Blundel, Woody and myself. Our changes are filtered back to Russell and from there back up to the mainstream kernel. While it is not presently the case, the NetWinder specific code will eventally be merged in completely with Russell's code which in turn be merged in to the mainstream linux kernel.

For now, the best place to get NetWinder-specific kernels is from netwinder.org (either via FTP or CVS). Russell's kernel (patches applied against mainline) also work but are not tested as widely, and may lack some drivers. Work is in progress to merge the sources together so that in the future there will be only one arm-linux source.

3.2 Getting the source

The source code for the NetWinder kernel can be obtained by anonymous FTP from `<ftp://ftp.netwinder.org/pub/netwinder/kernel/>`. The filenames are in the form `linux-YYMMDD.tar.gz` (consult the `README` file in the same directory for more information).

Alternatively, you may also download a binary RPM containing the source code (`kernel-source-YYYYMMDD.armv4l.rpm`). If you chose that route, you'll also need the corresponding `kernel-headers` package from the binaries directory. (There is also a `kernel-YYMMDD.src` package; this is only of use if you need to rebuild the kernel in RPM format and redistribute the results. Ignore it).

The kernel source normally lives in `/usr/src` but can really be installed wherever you wish. You should ensure that the symbolic link `/usr/src/linux` points at the place where your kernel source is located. The appropriate commands to unpack the source from `tar.gz` form and to make the symlink are shown below.

```
tar zxvf linux-YYMMDD.tar.gz
ln -sf /path/to/linux-YYMMDD /usr/src/linux
cd linux-YYMMDD
```

If you install the `kernel-source` and `kernel-headers` RPM packages, the source will be placed in `/usr/src/linux-YYYMMMDD` and the symbolic link will be set appropriately.

3.3 Configuring the kernel

As there are many ARM platforms, of which NetWinder is but one, the usual default configuration (`arch/arm/defconfig`) is not optimal for the NetWinder. In fact it might not even work. Instead, the default configuration can be found at `arch/arm/def-configs/netwinder`. If you're building a kernel tree for the first time, you should always copy the proper config file to the toplevel first:

```
cd <your-kernel-tree>
cp arch/arm/def-configs/netwinder .config
```

Alternatively, you can use the 'Load alternate configuration file' provided by `make menuconfig` or `make xconfig`, instead of copying the default configuration file as shown above.

The kernel must then be configured in one of the usual ways. The recommended way is via `make menuconfig`, but you can also use `make xconfig`, `make config` or `make oldconfig` (the latter is useful if you don't want to change any settings). Note that the `xconfig` option was broken prior to the 19990121 release.

For the first build, you *must* run one of the config tools (`menuconfig`, `xconfig`, etc) and answer "Yes" when it asks if you wish to save your configuration changes - even if you didn't change anything. Otherwise certain files will not be remade and the compile will fail later on.

The default configuration includes a minimum number of devices 'built in', and virtually everything that could be needed built as a module. For most cases, this should be sufficent. Of course, you can signficantly reduce the compile time by turning off options you won't need (for example, the whole SCSI subsystem, which is normally built as a module).

The bare-bones config for a NetWinder would normally include support for the '553 disk controller (unless you have no disk), have an `ext2` filesystem, and ethernet support for the `ne2k` (for eth0) or `tulip` (for eth1). There is a config called `netwinder-tc` intended for use on diskless NetWinders, which is pretty well stripped down.

The char driver for `ds1620` (thermometer) should always be included, since it operates the speed control for the NetWinder's fan. Otherwise, the fan might never turn on, and the NetWinder might overheat!

3.4 Building the kernel

Kernel compilation takes about 10-15 minutes on a stock NetWinder.

3.4.1 For kernel 2.0.x

For the 2.0.x kernels, the following commands should be executed.

```
make dep
make clean
make
make modules
```

Subsequent builds from the same source directory can (and should) omit the `make dep` and `make clean` stages.

The end result of this should be a kernel called `vmlinux` and a series of loadable modules in the `modules` directory, all within the build tree.

3.4.2 For kernel 2.2.x

For the 2.2.x and beyond kernels, the following commands should be executed.

```
make dep
make clean
make Image
make modules
```

Subsequent builds from the same source directory can (and should) omit the `make dep` and `make clean` stages.

The end result of this should be a kernel called `arch/arm/boot/Image` and a series of loadable modules in the `modules` directory, all within the build tree.

Compressed kernels also work (I forget exactly when, sometime around Dec 1999). Substitute `make zImage` instead of `make Image` if you wish it. The kernels are smaller, but take longer to decompress, so it is really only worth it if you are booting via tftp or similar. The output appears as `arch/arm/boot/zImage`.

Note that `bzImage` is not supported nor will it ever be! Contrary to the popular belief, `bzImage` does not mean a kernel compressed with `bzip`. It means 'big zImage', which has repercussions on how LILO loads the kernel on an x86 (with it's archaic 640k memory limit). The Netwinder doesn't have such a limitation, so we don't need `bzImage`.

3.5 Installing the kernel

To install the kernel, you must be logged in as `root`. It is a good idea to back up your existing kernel and modules so that you can revert to them later on, in case your new kernel doesn't work. To make the backups, use the following commands, substituting for `2.X.Y` with the actual version of your kernel.

```
cp /boot/vmlinux /boot/vmlinux.old
mv /lib/modules/2.X.Y /lib/modules/2.X.Y-old
```

Now the new kernel and modules can be installed. Note that the file `/boot/vmlinux` is usually a symbolic link. We recommend you give each kernel a unique name, and use the symbolic link to select which kernel to boot. Supposing you select the extension 'test' to identify your kernel, the following commands could be used.

```
cp arch/arm/boot/Image /boot/vmlinux-test
ln -sf vmlinux-test /boot/vmlinux
cp System.map /boot/System.map-test
ln -sf System.map /boot/System.map-test
make modules_install
```

The example above renames the kernel file from `Image` to `vmlinux-test`. This is not strictly necessary, all that counts is that the symlink points to the real kernel. You could theoretically point it into your build tree, but that would not be wise since a `make clean` would wipe it out (and then you'd be in trouble when you tried to reboot...).

3.6 Booting the new kernel

To test out the new kernel, you must reboot your machine. If you followed the instructions as described previously, your new kernel will (most likely) be booted automatically. This is because the default settings in the firmware are to load the kernel from a file called `/boot/vmlinux`, and you created a symbolic link to the actual kernel above.

If you used a different name for your kernel, or if your firmware settings have been changed from their default values, then you'll have to check and possibly modify the firmware settings. See the Firmware-HOWTO for more details about the firmware. In brief, you press a key when prompted while your machine boots. Then you issue the following commands at the firmware command prompt.

```
setenv kernfile /boot/YOUR-KERNEL-FILE
save-all
boot
```

Only specify the `save-all` command if you wish to make the setting permanent. Otherwise `kernfile` will default back to whatever it was before the `setenv` command. See the Firmware-HOWTO for more details.

3.7 If it won't boot

If your new kernel won't boot, then you probably forgot some important hardware support when you built it. To get back to a working system, you simply have to set the `kernfile` parameter in the firmware menu back to the old kernel `/boot/vmlinux.old`. What if you didn't make a backup? Well luckily the kind creators of the NetWinder have anticipated this need and provide a `/boot/vmlinux.rescue` kernel for this purpose. At least on the recent disk images.

If you can't determine why your new kernel won't boot, try using the default configuration (stored in `arch/arm/def-configs/netwinder`) when building the kernel. If that doesn't work, you're either doing something wrong, or using the wrong tools (compiler, binutils). All of the official NetWinder kernels to 20000121 were natively built using gcc-2.8.1 on an unmodified DM-13. (Future kernels will be built on dm-3.1-15 using gcc-2.95.1).

4 Other information

This chapter documents some known issues with the kernels, particularly the recent ones. I don't know how useful it will be to most readers, but for lack of a better place, this info is here. It also tells you how to get a cutting-edge(tm) kernel from our CVS repository.

4.1 CVS repositories

Those of you who really want to live on the cutting edge can retreive the latest kernel from the CVS repository on netwinder.org (module name is 'armlinux'). This source is not guaranteed to be stable or even buildable, and it could have lots of nasty side-effects (for example, if you tried to update your flash using version 2.2.7, it would erase the first block and then crash - leaving the machine unbootable). Please don't bug the developers about these kernels - use an officially blessed kernel!

The 'armlinux' module contains two main branches - one is called 'armlinux-2_2' and holds the 2.2.x kernel development, while the other (main) branch contains 2.3.x development. To get the latest stable 2.2.x kernel you'd use these commands in `bash`

```
export CVSROOT=:pserver:USERID@netwinder.org:/cvs
cvs login
  ... enter your password for netwinder.org ...
cvs co -rarmlinux-2_2 armlinux
```

Omit the `-rarmlinux-2_2` tag to retrieve the current 2.3.x kernel tree.

If you don't have a developer account, you can register for one on the web site. (In the future we'll use SSH for write-access and there will be an anonymous pserver account for retrievals, right now you have to register).

4.2 Firmware requirements

The 2.2 kernel series, and for that matter, the 2.0.35 series after 99/01/21, use a new method for receiving boot-time arguments. In the old days, all the settings (such as which device to use for root filesystem) were passed on a command line - just a plain text string. This is a pain to parse and as the number of options grows, so must the text buffer. The new method defines a structure where the data can be encoded in binary (rather than plain text) which makes parsing much quicker and there's lots of room to grow. Like it or not this "param struct" is here to stay.

What this means practically is that you need to run firmware which knows of the parameter structure, and passes a properly initialized structure to the main kernel. If you use older firmware, the kernel will not recognize the structure, and will revert to a default command line instead. Most of the time the default command line isn't right (for example, it might say you only have 16 MB of RAM, when you have more, or it might try to boot from the wrong partition). The values you select in the firmware won't get passed to the kernel in this case.

So which firmware should you use? Something after 2.0.8h, such as 2.1.16 or 2.1.24, are the recommended choices. Have a look at `<http://www.netwinder.org/~ralphs/compat.html>` for more details on the various firmware versions.

4.3 Using 2.2.x on older builds

If you install a 2.2 kernel onto an older build (dm-12 for example), a number of user-space services won't work right anymore. First off, consult the file `Documentation/Changes` in the kernel source - it points out most of the things that will go wrong.

Many of the daemons (ftp, telnet, etc) need to be recompiled before they work. Some need to be updated to newer versions. It's kind of ugly... Much easier would be to get the current developer disk image (dm-3.1-15) or the base image (base-3.1-15) and install that instead.

Older X servers won't run with the 2.2 kernel. A more recent version of the SVGA server can be fetched from netwinder.org, and you'll need to create a device node `/dev/fb0current` device node with major=29, minor=0. Alternatively, the old XF68_FB server can be used as well (just remember that it pretty much ignores etc/XF86Config).

Also, the `/dev` entries must be updated. Most notably, tty and pty entires are needed so that remote login (and xterms) can run. Take a newer /dev/MAKEDEV script and run it on the NetWinder.

Ultimately, I hope to have convinced you to just download the dm-3.1-15 image if you've still got a system with kernel-2.0.x on it. If you cannot download it, you can send me a few dollars to cover my costs and I will send you a CD (please, only ask if you have no other way of getting it!).

4.4 `tcp_ipv4_rechecksum`

If you see a lot of the above message in your `/var/log/messages` file, rest assured there is nothing wrong. It's a piece of debugging code that I forgot to take out, because I thought it wouldn't happen very often. Turns out that any decent amount of ethernet traffic causes the error. It will go away in a future version.

4.5 `Buggy processor` or `Wierdness detected`

This is a warning that Russell put in to try and track a suspected bug in the CPU. Both messages refer to the same condition; it was reworded because too many people got worried about the "buggy" part of it. Russell has a nice description on his website, and the error message will direct you there if you see it. Essentually, the CPU is flagging an error condition that it can't possibly have encountered, so the suspicion is that it's a bug in the silicon.

The (suspected) bug is harmless (other than the message) and will not cause any programs to burst into flames or otherwise fail. If you see the message (especially, if you can reliably reproduce it) then please mail the details to Russell and myself. Otherwise just ignore the message if you see it.

4.6 `trying to free non-small page`

This message occurs frequently upon shutdown of an officeserver system running the latest 2.2.13 kernel. It's related to virtual memory system, and it kind of scares me, but so far I've not been able to track down what it really means. I've also not seen any adverse effects (it only happens on shutdown, and the system starts up fine despite it). Hopefully it will just "go away"...

4.7 `task can't get free slot`

This message is generated by the NFS subsystem from time to time (usually when there is or has been extended periods of network trouble). Not sure what to do about it at this time.

5 Misc

5.1 Author

The author and maintainer of the NetWinder Kernel-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/Kernel-HOWTO.html>`.

5.2 To-do

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?

5.3 History

April 29, 1999 (version 1.1): First public release.

May 22, 1999 (version 1.2): Updated 2.2 kernel information.

June 4, 1999 (version 1.3): Added section about firmware and 2.2

June 10, 1999 (version 1.4): Corrected URL for kernel source... doh!

Sep 3, 1999 (version 1.5): FTP path corrected from ccc to netwinder.

Feb 15, 2000 (version 1.6): Major rewrite for 2.2 kernels and beyond.

5.4 Contributors

5.5 Legal stuff