February 2000
DeveloperTechNote-6 Revision 1-1

Adding a Kernel to the RaQ 3 Server Appliance Family

1 Overview

The Cobalt RaQ 3 ROM will always load a single Linux kernel. Developers who build their own Linux kernels need a spare one in case the kernel that they build is defective. This Technical Note tells you how to add your kernel to the RaQ 3 using the Boot from ROM (`bfr`) and Boot from Disk (`bfd`) commands.

Table of Contents

Technical notes are located in `ftp://ftp.cobaltnet.com/pub/developer/`

1.1 Audience

This Technical Note is intended only for software developers creating or modifying Linux kernels to be used with any of Cobalt Network's RaQ 3 server appliances. Most software developers will use the Linux kernel that ships with the Cobalt product, and will not need to build one themselves. Also, since the existing kernel fully supports the Linux module mechanism, many desired new features, such as plug-in device drivers, can be added as modules to the existing kernel.

Note
Replacing the Linux kernel voids your Cobalt software warranty. Cobalt strongly recommends that you build and test your kernel on a nonproduction system before installing it on a mission-critical system.

1.2 Applicable Products

This Technical Note applies only to the RaQ 3 server appliance family.

2 Adding a Kernel on the RaQ 3 Server Appliance Family using the ROM Interface

To get a ROM monitor interface on your terminal emulator to the RaQ 3:

1) Attach a serial line to the console port, which is the left serial port on the back of the RaQ3.
2) Using a terminal emulator program of your choice, for example, `minicom` in Linux, `SmartModem` or `HyperTerm` in Windows, or `zterm` in Mac OS, set the port configuration to 115200 baud, 8 bits, no parity, 1-stop bit.

   In order to maintain your existing RaQ 3 functionality, you must obtain the kernel modules that match the kernel you are about to build. This is necessary because `insmod` requires that the kernel and module version match. This is particularly necessary if you want to use the Cobalt bandwidth management module, since tit is only available in binary format. Get the latest kernel from the Cobalt FTP site:

```
ftp://ftp.cobaltnet.com/pub/products/raq3/RPMS/kernel-2.2.12C5-1.i386.rpm
ftp://ftp.cobaltnet.com/pub/products/raq3/RPMS/kernel-headers-2.2.12C5-1.i386.rpm
ftp://ftp.cobaltnet.com/pub/products/raq3/RPMS/kernel-source-2.2.12C5-1.i386.rpm
```

```
ftp://ftp.cobaltnet.com/pub/products/raq3/RPMS/bwmgmt-1.0-6.i386.rpm
```

Note
The above versions are current as of the writing of this Technical Note, but a more up-to-date version might be available.

3) Install the modules using the `U` upgrade option to the RPM command:

```
% rpm -hUv --force kernel*2.2.12C5-1* bwmgmt-1.0.6.i386.rpm
```

Build your kernel:

1) Check to see if there is a newer kernel for the RaQ 3 on from Cobalt's web site located at `ftp://ftp.cobaltnet.com/pub/experimental/`

   Cobalt updates the kernel as necessary. The latest experimental kernel releases can always be found at the above URL. The most recent `bwmgmt` package needed for each kernel revision is also located at the above URL.

2) Change directories to the kernel source directory. Type:

```
cd /usr/src/linux
```

3) Build a kernel configuration file. Type:

```
make oldconfig
```

4) Prepare a base kernel to verify your build environment. Type:

```
make dep clean
```

5) Build the kernel image. On the Cobalt-supplied kernel source, type:

```
make cobalt
```

   On the non-Cobalt-supplied source, type:

```
make vmlinux
gzip -f -9 vmlinux
```

   This build will take approximately 10 minutes to complete. This should build a kernel with the same functionality as the binary RPM. Checksums will not match, because of embedded date/time strings in the kernel image.

Note
If you are familiar with Linux on standard personal computers, you might be familiar with the `make zImage` or `make bzImage` commands. Do **not** use these commands. Cobalt systems do not need the extra information in these formats, and these commands do not work.

6) Make the changes you are planning to the kernel source tree.

7) If your changes did not involve changing the kernel build configuration (using either `make config`, `make menuconfig`, or `make xconfig`), repeat Step 3. If your changes did involve building the config file, proceed to the next step.

8) Repeat the commands in steps 4 and 5. You should now have a kernel image that reflects your changes.

9) Copy the new file to a new file in /boot. Type:

```
cp vmlinux.gz /boot/mykern.gz
```

After you have built your kernel, follow these steps:

1) Reboot.

2) Press the space key from the terminal emulator window during the boot process after you see the ROM banner. You should see a prompt:

```
Cobalt:Main Menu>
```

3) To boot your kernel, type:

   bfd /boot/*mykern.gz*

4) When the system has finished booting, verify that the kernel that is running is the correct version by looking at the date in the output of the following command:

```
uname -a
```

5) Save the old kernel, make the new one the default. Type:

```
cp /boot/vmlinux.gz /boot/cobalt-vmlinux.gz
cp /boot/System.map /boot/cobalt-System.map
mv /boot/mykern.gz /boot/vmlinux.gz
cp /usr/src/linux/System.map /boot/System.map
```

Note

If you ever need to boot the original Cobalt kernel, repeat steps 1 through 5 above, substituting the name `/boot/cobalt-vmlinux.gz for /boot/mykern.gz`.

The */usr/games/.doug* backup kernel from Cobalt's MIPS-based products does **not** work on the RaQ 3.

3 Booting From ROM

If needed, you can also boot the system using the `bfr` (`boot from ROM`) command to the boot menu; see steps 1 through 5 above. This brings the system up running the kernel that is in ROM.

You can also boot the ROM kernel from the front panel:

1) While booting, when the Cobalt logo is scrolling, press and hold the **Select** button until the LCD displays `Select option`:

Note

Press the **S** button for **Select**. Do **not** press the four directional buttons.

2) Use **Select** to scroll to `Boot from ROM`.

3) Press **Enter**.

You will need the hard disk in the Cobalt server for the file system, but the system will boot from a known good kernel, even if the kernel you compiled fails to boot.