February 2000
DeveloperTechNote-7 Revision 0-0

Adding a Kernel to Cobalt Networks' MIPS-based Server Appliance Family

1 Overview

The Cobalt MIPS-based Qube or RaQ 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 Qube and Qube 2, and RaQ 1 and 2 family of server appliances.

Table of Contents

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 Qube and Qube 2, RaQ 1 or 2 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.

For information on debugging Cobalt kernels, see `ftp://ftp.cobaltnet.com/pub/users/cjohnson/debug/README`.

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 Qube and Qube 2, and RaQ 1 and 2 server appliance family. For information on creating or modifying kernels for Cobalt's RaQ 3 family server appliances, see DeveloperTechNote6-0-0.pdf located at `ftp://ftp.cobaltnet.com/pub/developer;` all Technical Notes are located in `ftp://ftp.cobaltnet.com/pub/developer/`

2 Adding a Kernel to the Qube or MIPS-based RaQ using the ROM Interface

Before adding a kernel, make sure you have downloaded the latest kernel source from the Cobalt web site located at `ftp://ftp.cobaltnet.com/pub/experimental/`. This kernel will support modules and works for all of Cobalt's MIPS-based servers. Also, get both `All-Kernel_mips_tools-1.0.pkg` and `All-Kernel_mips-1.0.pkg` from `ftp://ftp.cobaltnet.com/pub/developer`.

To get a ROM monitor interface on your terminal emulator to the Qube or MIPS-based RaQ:

1) Attach a serial line to the console port.

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

Build your kernel:

1) Get the source Redhat Package Modules (RPMs) from Cobalt's web site located at `ftp://ftp.cobaltnet.com/pub/experimental/`.

```
ftp://ftp.cobaltnet.com/pub/products/raq3/RPMS/kernel-2.0.34C51_SK-2.src.rpm
ftp://ftp.cobaltnet.com/pub/products/raq3/RPMS/kernel-headers-2.0.34C51_SK-2.rpm
ftp://ftp.cobaltnet.com/pub/products/raq3/RPMS/kernel-source-2.0.34C51_SK-2.rpm
```

Note
`kernel-2.0.34C51_SK-2.src.rpm` is the current MIPS kernel, but it might be updated. Use the kernel with the highest number. The latest experimental kernel releases can be found 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 takes approximately 30 minutes to complete. This should build a kernel with the same functionality as the binary RPM; however, checksums will not match, because of embedded date/time strings in the kernel image.

Note
If you are familiar with Linux on personal computers, you might also be familiar with the `make zImage` or `make bzImage` commands. Do **not** use these commands. Cobalt servers 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 the server.
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:
```

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

3 Booting the Backup Kernel from the Front Panel

If you overwrite the default kernel, `/boot/vmlinux.gz`, with a kernel that does not boot, there is a backup kernel located on the disk for emergencies.

While booting, press the four directional buttons at the same time and hold them down. The booter will load the kernel from `/usr/games/.doug`. This file should be a `gzip` compressed kernel, just like the one in the `/boot/vmlinux.gz` file.