Kernel Debug Task

Compile and debug the Linux kernel on the Accelent system.

Please read the entire document before starting. While readying this document you will find sections that are narrower than the regular text and enclosed in a box. These sections are informative about exceptions to the "normal" steps to take through this task and may contain corrective steps that can be taken under certain circumstances.

There are two Linux systems involved in this procedure, the host and the target. The host system is used to edit, compile and debug the kernel cross-compiled for the target. The host in this case is an Intel based PC running Red Hat Linux 7.2 The target is a strong-arm based embedded system running Linux 2.4.4.

A tool chain has already been installed on the host system that allows kernel source code on the host to be cross-compiled to the target machine. A tarball found in `/home/class/kernel.tar` with the complete kernel source and all of the current patches is provided to build and debug Linux on the target machine.

The kernel tarball can also be found on millserv (at 129.219.204.69) in `/home/class`.

Begin by opening a terminal using the command panel at the bottom of the screen. Now create the kernel directory in your home directory. Enter the command:

tar -xf /home/class/kernel.tar

This should create a Linux kernel source tree in your home directory.

Change to the kernel directory and list the Linux directory and the patch files that were already applied to this kernel as follows:

cd kernel
ls -l

It is important to clean the kernel build when making the transition between a debug and a non-debug kernel. Use the following commands to do this.

cd linux
make clean

The kernel has to be configured before it can be built. This is done with the following commands and data entry. First type:

make menuconfig

A menu will be displayed that allows the Linux kernel to be customized. Using your arrow keys (the mouse doesn't work in this menu), scroll to near the bottom of the menu (the next–to-last line) and select:

Load alternate configuration file

by pressing the Enter key.
You will then be prompted for a configuration file name, you need to enter

debug.config

However, there will likely already be some extraneous characters in the input field. You must get rid of them. Backspace as much as you can. If there is some text still in the text field press the tab character multiple times until the cursor returns to the text field. You then type in the file name given above (at the beginning of the line or after any characters that could not be deleted) and press the Enter key again to return to the master menu.

This will load all of the configuration settings for the new kernel. These are required to define a Linux kernel that can run on the Accelent system and be debugged remotely from the host. When you next "make" the Linux kernel (see the next steps), you will start the cross-compile, -link and -load steps to create a "debug-able" kernel. (The make boot_image step changes the "load" address of the new kernel to the address needed for the taget achitecture.)

You now will leave the menuconfig program. Use the right or left arrow key to highlight Exit at the bottom of the window and press the Enter key again. As you leave menuconfig you will be prompted to save the configuration. Be sure to select Yes and do this step (i.e., save the configuration settings).

Now that the kernel has been properly configured you may build it using a series of make commands. Each make command may require several minutes to complete:

make dep
make zImage
make modules
make modules_install
make boot_image
ls -l

The first time through you should do this one line at a time just to see how long each step takes. One subsequent iterations, you may include all of these on one line by separating each command by a semicolon. Also, if you are remaking a debug kernel for the second time in a row, you don't have to do a make clean nor some of the other make steps, which ones will be covered later.

The kernel debug information will have been stored in the file `vmlinux.dbg`. It should be over 6MB in size. If it is smaller you probably didn't successfully load the `debug.config` in the **make menuconfig** step or you didn't clean the build before you started. Go back to **make clean** and try again.

When you have successfully built the kernel there will be a file named `nk.bin` in the `kernel/linux` directory (this is where you should be at this point). This file contains the cross-compiled-linked–loaded kernel object. It must now be transferred to the Accelent system flash card in order to boot it. Please log onto the Accelent system using `seyon`. Open a second terminal window and type the following command:

seyon

In the `seyon` terminal window press Enter then enter the following in response to the login and password prompts from the Accelent Linux system.

root
root

If the Accelent system was not at a login prompt or is not responsive after you hit Enter you're advised to reset the Accelent by pressing the reset button, the small black button located behind the post nearest the right side of the Accelent keypad.

If the Accelent remains unresponsive: You may need to mount the flash card from the Accelent console and erase it. Typing on the Accelent keyboard use the commands:

cd /root
mount /dev/hda1 hda1
cd hda1
ls

This will show any files left by the last user. Erase them.

rm *
sync

Now reset the Accelent again. And go back to the `seyon` step.

After logging in mount the flash card and change directory using the following commands:

mount /dev/hda1 hda1

If /dev/hda1 is already mounted proceed to the following command:

cd hda1
ls

If any files are present on the flash card erase them.

rm *

Connect to the host via ftp:

ftp 10.0.0.1

This may take 2 or three minutes the first time you try connecting to the host from the Accelent target. Once connected enter your user name and password when prompted for Name and Password.

When logged-in to the host via ftp, type:

cd kernel/linux
get nk.bin

This will retrieve the debug`able kernel image that you built and placed in the kernel/linux directory on the host.

When the transfer has finished enter:

quit

to exit ftp.

Remember to sync or you may lose the new kernel, i.e., memory buffers on the Accelent may not be flushed to the compact flash ‘disk’.

sync

You are now going to reboot the Accelent system with the debug kernel you just built and downloaded. First, close the `seyon` session (by clicking x in the upper right hand corner of the Seyon window) so the serial port can be used by the debug session. Now reset the Accelent system (use the reset button). The Accelent bootloader looks first for the file `nk.bin` on the flash card for a kernel image. When found it boots by loading the kernel image on the flash card. By now you should have successfully booted your new Linux kernel. If so you will see the usual objects appear on the Accelent display panel.

> Improperly mounting the flash card (using mount) and downloading the kernel image elsewhere may result in some subtle problems. We have noticed that some people are saving their work to the directory /hda1 or to /root/hda1 with no mounted flash card. This causes the files transferred by ftp to be stored in the on board Accelent flash memory. While this will work for applications, it won't allow you to boot your new kernel. Instead it will boot the default kernel or a kernel left on the flash card by someone else. You may spend hours wondering why your kernel changes never appear or the kernel won't debug.

If everything works properly we can now try to debug the kernel.

On the host you must be in the build directory to debug. Use `pwd` to confirm your current working directory:

pwd

This should result in `/home/"your account"/kernel/linux`. If it is not correct enter the following command:

cd $HOME/kernel/linux

Start the debugger:

gdb-arm vmlinux.dbg

This will start an interactive session on the host with the host cross-debugger using the debug data stored in `vmlinux.dbg`

Use the following sequence to connect the host debugger to the target debugger:

From the debugger **File** menu choose **Target Settings.**
Choose **Remote serial** as the **target connection.**
Enter **115200** as the **connection baud rate.**
Enter **/dev/ttyS1** as the **connection port.**
Open the **'More Options'** section**.**
Disable **'Download Program'** by making the button a non-yellow **'outie'.**
Choose **OK.**
Open a console window by typing **ctrl-n.**
From the **Run** menu choose **Connect to Target**.

> If you get a message about a shared library handler failing, just ignore it for now. You do this by clicking OK in the pop-up message windows.

You should now be successfully connected.

The kernel will stop in the breakpoint function. If not click on the Stop sign. Click on the running man to start it again. You will notice all of the boot messages appearing on the debug console window you opened earlier. You may click on the Stop Sign to break into the kernel again at any time. At this point you may set other breakpoints. You should try stepping through the kernel and setting other breakpoints. When you are finished, to exit the debugger - first break into the kernel (hitting the Stop Sign to get the Running Man) and close the debugger by hitting **exit** on the **file** menu. This enables the kernel to continue to execute.

> Once the degugger has halted you will find that `seyon` does not work with the debug version of the kernel. So you must work interactively with your debug kernel by using the terminal on the Accelent display and the Accelent keyboard.

Please remove `nk.bin` from the flash card before leaving. You must do this through the Accelent console. Be sure to `cd` to /root before mounting the flash card. You must remount each time (after) you reboot the Accelent.

> If the Accelent remains unresponsive at any point above, you can, as a last resort recovery method, remove the flash card about ½" and reset the Accelent again. Push the flash card back in after the reset is complete, mount the flash card and erase it. Please do not do this unless you absolutely have to.

> If you cannot return the systems (target or host) to the state they should be for the next user, go to a computer that will let you e-mail to Dr. Millard and send him a message about the problem. His e-mail address is Bruce.Millard@ASU.EDU.