Homework 1

CS444 Spring2018

Austin Hodgin, Joshua Novak, Trent Vasquez

April 30, 2018

Abstract

Description on how to build the kernel and run it in a virtual machine on the OS2 server, as well as all the appropriate flags to call in qemu.

1 COMMAND LOG

For our commands we ran two different terminals connected to os2, one to setup the kernel and check things, another to run gdb. These will be broken up into separate sections to make the process easy to understand. For commands to the terminal doing work to setup the kernel, the section will be labeled Kernel, for the terminal running GDB the sections will be labeled GDB.

1.1 Kernel 1

- git clone -n https://git.yoctoproject.org/git/linux-yocto
- cd linux-yocto
- git checkout tags/v3.19.2
- cp /scratch/opt/core-image-lsb-sdk-qemux86.ext4 ./
-
- cp /scratch/opt/core-image-lsb-sdk-qemux86.ext4 ./
- cp /scratch/opt/bzImage-qemux86.bin ./
- source /scratch/files/environment-setup-i586-poky-linux.csh
- qemu-system-i386 -gdb tcp::5524 -S -nographic -kernel bzImage-qemux86.bin -drive file=core-image-lsb-sdk-qemux86.ext4,if=virtio -enable-kvm -net none -usb -localtime –no-reboot –append ”root=/dev/vda rw console=ttyS0 debug”

1.2 GDB 1

- gdb -tui
- (return)
- target remote tcp::5524
- continue

1.3 Kernel 2

- root
- exit
- (ctrl+a)
- (c)
- quit
- cp /scratch/opt/config-3.19.2-yocto-standard ./.config
- make -j4 all
- make menuconfig

1.4 Navigate UI

For this section I navigated the menu as follows to make a change to the config file which I could identify later. First I navigated down the menu to General setup and pressed enter. Then I navigated down to the line that begins with (-yocto-standard) which is the name of the configuration and pressed enter. This brought me to an area where I could edit the name of the configuration. I added the characters ”Group24HW1” and pressed enter (the string in the quotes, not including the quotes). Then I pressed right to get to exit and pressed enter. I pressed right to get to exit again and pressed enter. I then pressed enter at the prompt to save my configuration.

1.5 Kernel 3

- make -j4 all
- qemu-system-i386 -gdb tcp::5524 -S -nographic -kernel arch/x86/boot/bzImage -drive file=core-image-lsb-sdk-qemux86.ext4,if=virtio -enable-kvm -net none -usb -localtime –no-reboot –append ”root=/dev/vda rw console=ttyS0 debug”

1.6 GDB 2

- target remote tcp::5524
- continue

1.7 Kernel 4

- root
- uname -r
- The return value from this should be 3.19.2-yocto-standardGroup24HW1

2 FLAGS

- qemu-system-i386
  - Calls the installation of Qemu
- -gdb tcp::????
  - Sets the instance to be able to be debugged with gdb at the address tcp::????
- -S
  - Has the instance halt immediately upon starting so that it can be debugged
- -nogrphic
  - Has the instance run in graphic-less mode
- -kernel bzImage-qemux86.bin
  - Has the instance of Qemu run using the kernel buit with name bzImage-qemu86.bin
- -drive file=core-image-lsb-sdk-qemux86.ext4,if=virtio
  - Has the instance of Qemu use the file core-image-lsb-sdk-qemux86.ext4 as a drive. The if=virtuo part of the argument has it be used as a virtual drive.
- -enable-kvm
  - Enables Kernel based virtualization support.
- -net none
  - Configures a host network backend. The none flag configures the instance of Qemu such that there is no host network.
- -usb
  - Enables the USB driver
- -localtime
  - Sets the time to run in local time. This is a depreciated flag as far as I can tell, and is instead run as an option of the -rtc argument
- –no-reboot
  - Makes you exit instead of rebooting.
- -append ”root=/dev/vda rw console=ttyS0 debug”
  - Directs the instance to us ”root=/dev/vda rw console=ttyS0 debug” as the command line

3 QUESTIONS

3.1 What Was The Point?

Showing that we can build our own instance of the Kernel correctly as well as teaching us about the flags for Qemu

3.2 Approach

We followed the listed instructions and asked TA’s, friends who had taken the class before, and Kevin for help when necessary.

3.3 How Did You Ensure Your Solution Was Correct?

We made a change to the configuration’s label and saved it to our config file. We then launched the kernel in Qemu and checked that the change to the label showed up when using uname. If the change wasn’t there, then we would have known that the instance of the kernel we launched was built using the config file we changed, meaning it was likely the image given to us. Either that, or we did not build using the correct config file.

3.4 What Did You Learn?

We learned how to compile the kernel on the engineering server, the functions of various flags for the Qemu launch command, and how to connect to a virtual instance of the kernel in GDB. We also learned a lot about git. I imagine that will be a regular occurrence for this class.

4 Version Control Log

Date	Author	Description
04-12-2018	AustinHodgin	Initial commit
04-12-2018	AustinHodgin	Added .gitignore and linux-yocto
04-12-2018	AustinHodgin	Fixing linux-yocto not being added
04-12-2018	AustinHodgin	Trying to add linux-yocto/ and fixed .gitignore
04-12-2018	Joshua Novak	Trying to commit again
04-12-2018	Joshua Novak	Removed nested repo

5 Work Log

5.1 Tuesday

Austin and Joshua met and worked on downloading the git repo for two hours, were able to create the folder, download the repo, and launch the provided image of the kernel.

5.2 Wednesday

Joshua and Trent worked on building the Kernel for an hour and a half. They were able to build and launch the kernel.

5.3 Thursday

Joshua ran the necessary commands to launch the kernel and prove that the new instance of the kernel was running and recorded them. This took roughly thirty minutes. Austin and Joshua worked on the write up and fixing the repo. This took roughly two hours.