CS444 Homework 1

Lilliana Watts, Nicholas Skinner, Yong Ping Li

April 15, 2018

COMMANDS USED

Command log: From directory Spring2018/group7/yocto-linux-3.19.2

Terminal 1: make mrproper

cp ../config-3.19.2-yocto-standard .config

make menuconfig

make -j4 all

source ../environment-setup-i586-poky-linux

qemu-system-i386 -gdb tcp::5507 -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”.

Terminal 2: source ../environment-setup-i586-poky-linux

$GDB

target remote localhost:5507

continue

Terminal 1:

(when prompted) root

uname -r

We confirmed we were running the right kernel with the following method:

1. Upon calling make menuconfig with terminal 1, changed the Local Version from yocto-standard to group07kernel. 2. After booting the kernel with qemu, using the command uname -r to print the local version. Upon seeing it was ”group07kernel,” we know it is correct.

QEMU FLAGS

A. Command Line:

-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”.

Individual flags:

•-**gdb**: This flag is followed by a device (for this example localhost 5507) and signals a wait for a gdb connection on the device.

•-**S**: this tells it to not start CPU on startup. This is the reason why the command will halt until you type continue.

•-**nographic**: This disables the graphics, and makes qemu into a command line interface.

•-**kernel**: This signals to use the file that follows as the kernel image.

•-**drive**: This defines a new drive to use. The drive is used with the disk image that follows after the flag, for this setup, that is file=core-image-lsb-sdk-qemux86.ext4

•-**enable-kvm**: This flag enables full kernel based VM support, allowing us to use the linux kernel to create and run a VM.

•-**net none**: No network devices are being configured with our setup, so we use this flag.

•-**usb**: This flag enables the usb driver. We can use usb to upload files to the VM.

•-**localtime**: Boots up the kernel with the current UTC.

•–**no-reboot**: this flag makes it so that instead of allowing a reboot, the program will just exit.

•–**append**: This flag passes command line arguments to the booting kernel, it appears that in this example it is located in /dev/vda, it is using ttyS0 for the linux port, and the kernel is starting in debug mode.

CONCURRENCY QUESTIONS

B. What do you think the main point of this assignment is?

•For the concurrency, this assignment is intended to refresh our knowledge of parallel programming, by having us implement a multi-threaded program with shared resources.

•For the homework, this assignment is intended to prepare us for our future assignments by showing us how to setup and run our kernel.

C. How did you personally approach the problem? Design decisions, algorithm, etc.

•For the concurrecy, we first researched the information provided in the class resources, as well as the operation of mutex locks. The example provided by the book had been a useful starting point as it had a step-by-step approach. As the rdrand instruction is not supported on the provided os2 servers, we instead used the Mersenne Twister C Implementation[1] for our code. In our code, it was used by creating a function that generates a number within the given range passed to the function (two ints).

•For the Homework, we used the provided instructions and some basic googling to get our environment ready. No intensive programming was required, just a few command line arguments.

D. How did you ensure your solution was correct? Testing details, for instance.

•For the concurrency, Printf statements were used frequently within the program's operation, like when a producer had created an item, when a consumer removed and item, as well as when a buffer had been accessed. the program had been left to run until 100 items had been both produced as well as consumed. and the output had been checked to determine if that had what was expected.

•For the homework, we had to confirm that our kernel was running my manually starting and remoting into it.

E. What did you learn?

•We learned how mutex locks and pthread waits operate, and how to implement both correctly. More specifically, we learned how pthread cond signals are used to unblock the threads, and when to call them.
•We also learned some of the basics that we will need moving forward for future assignments.

GITHUB LOG

Detail	**Author**	**Description**
9138e68	Nicholas	Initial commit
c0384d2	Nicholas	Command Line Flags
05e52f4	Nicholas	FixFormat ConcQuestions
74e34ce	Nicholas	Add Table to Writeup

WORK LOG

Date	**Name**	**Hours**	**Description**
4/14	Nicholas	2	writeup
4/13	Nicholas	2	writeup
4/12	Lilliana	2	Kernel set up
4/12	Nicholas	2	writeup

REFERENCES

[1] Makoto Matsumoto and Takuji. Nishimura. qemu-system-x86_64(1) qemu-system-x86 debian jessie debian manpages. http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/CODES/mt19937ar.c. (Accessed on 04/13/2018).