CS 444 Assignment 1

Alessandro Lim, Kevin Turkington

I. Command List

1) cd /scratch/spring2017/
2) mkdir 13-07
3) git clone git://git.yoctoproject.org/linux-yocto-3.14 in 13-07 directory
4) git checkout v3.14.26 Change the version to v3.14.26.
5) source /scratch/opt/environment-setup-i586-poky-linux.csh
   Run the environment configuration script for the shell, which is required to run the qemu.
6) cp /scratch/spring2017/files/config-3.14.26-yocto-qemu .config
7) make menuconfig
8) /
9) LOCALVERSION
10) make -j4 all
    This function is used to build the kernel, and the j4 flag sets the thread used to 4.

    On the second terminal:
11) source /scratch/opt/environment-setup-i586-poky-linux.csh
    This is again used to set the environment configuration.
12) cp /scratch/spring2017/files/bzImage-qemux86.bin .
13) cp /scratch/spring2017/files/core-image-lsb-sdk-qemux86.ext3 .
14) qemu-system-i386 -gdb tcp::5637 -S -nographic -kernel bzImage-qemux86.bin -drive file=core-image-lsb-sdk-qemux86.ext3,if=virtio -enable-kvm -net none -usb -localtime –no-reboot –append ”root=/dev/vda rw console=ttyS0 debug”
    This starts the qemu VM in the terminal. Port 5637 is used as the port for the group.

    Back to the first terminal:
15) gdb
16) target remote:5637
17) continue
    This will connect gdb to the qemu. Typing continue will allow the VM to run on the other terminal.
    On the second terminal, the virtual machine should now be shown.
18) login as root, then type uname -a
    uname -a is used to print the system information.
19) reboot
    Since –no-reboot flag is set, the VM will shutdown instead of being reboot.
20) qemu-system-i386 -gdb tcp::5637 -S -nographic -kernel linux-yocto-3.14/arch/x86/boot/bzImage -drive file=core-image-lsb-sdk-qemux86.ext3,if=virtio -enable-kvm -net none -usb -localtime –no-reboot –append ”root=/dev/vda rw console=ttyS0 debug”

    Back to the first terminal:
21) gdb
22) target remote:5637
23) continue
    on the first terminal:
24) root
25) uname -a
26) reboot

II. Concurrency

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

The point of this assignment was to review material from CS344 as well as expanding on that material. Specifically how to create multiple threads in a C program and how they can interact with each other by signaling. In addition, this assignment was an introduction to incorporating assembly into our programs.

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

We begun by creating a basic framework of all the data structures, blank functions, and variables

that we thought would be need for the assignment. Afterwards we started reading man pages, and stack overflow threads on event driven programming and how to accomplish it. After we had a basic understanding of events, we started to research how to make threads talk to each other. From there it was a matter of creating the setup for structs to be created and deleted. Then passing access to the data back and forth between the producer and consumer threads.

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

First we tested the that the program created an exclusive lock to the struct holding all the data, this was done by checking if each thread would block after completing its task. After that was verified we check that each thread successfully signaled to each other when it was done producing or consuming. This was done by creating a collection of print statements unique to producers and consumers. And finally, since in our design that consumer doesnt actually delete data but instead move to the next node until the arrays maximum and wraps back around. We checked that when the producer wrapped back around that it overwrote old data. This was verified by reducing wait times and checking item numbers for duplicates when the prodcuer and consumers wrapped back to the front of our data array.

D. What did you learn?

From the concurrency assignment we learned the basic concept of event driven programming and how to implement it theoretically with the producer consumer problem. To expand on this, as for the technologies we used to achieve event based programming, we learned how to lock data to a particular thread using locking and unlocking. as well as how to make threads interact with each other through signaling and blocking (waiting for the mutex to be unlocked.)

III. QEMU COMMAND FLAGS

qemu-system-i386 -gdb tcp::???? -S -nographic -kernel bzImage-qemux86.bin -drive file=core-image-lsb-sdk-qemux86.ext3,if=virtio -enable-kvm -net none -usb -localtime –no-reboot –append ”root=/dev/vda rw console=ttyS0 debug”.

1) qemu-system-i386
   Begins a 32bit qemu session.
2) -gdb
   Creates opens a gdb-server on a specified port. In this case we are using port 5637
3) -S
   Disallows the CPU to start on boot.
4) -nographic
   Qemu is set to a command line interface and will not start a desktop enviroment like KDE, Cinnamon, or Unity.
5) -kernel
   Boots a kernel without installing the disk image.
6) -drive
   Indicates a specific drive for the Qemu instance. In this case a specific file is being used ”core-image-lsb-sdk-qemux86.ext3”. Additionally a specific interface is being used ”virtio -enable-kvm”
7) -net
   Prevents the kernel from configuring any network devices to this instance.
8) -usb
   Enables the use of the USB driver.
9) -localtime
   Specifies the local-time must be used for this instance.
10) –no-reboot
    If told to reboot, the instance will end instead.
11) –append
    Uses a specific kernel command at startup.

IV. GITHUB LOG

Detail	**Author**	**Description**
656b014	Kevin	Initial commit
97528f1	Kevin	adding semi completed concurrency
c878146	Kevin	finishing concurrency 1
7428a0c	Kevin	changing minor newline issue
4289f27	Kevin	ironing out overwriting issue, refactoring
412720e	Kevin	refactoring check for x86 system from class example
065c49d	Kevin	adding command line parameter

V. Work Log

Date	Name	Hours	Description
4/08	Kevin	2	Concurrency 1 setup
4/09	Kevin	5	Concurrency 1 research and writing
4/10	Kevin	2	Concurrency 1 refactoring
4/13	Kevin	1	Concurrency 1 refactoring
4/17	Kevin	2	writeup
4/18	Kevin	2	writeup
4/18	Kevin	3	Concurrency 1 refactoring
4/18	Alessandro	2	VM set up
4/18	Alessandro	2	writeup

References

[1] Linux manpages online - man.cx manual pages. https://man.cx/qemu-system-x86_64(1). (Accessed on 04/19/2017).

[2] qemu-system-x86_64(1) qemu-system-x86 debian jessie debian manpages. https://manpages.debian.org/jessie/qemu-system-x86/qemu-system-x86_64.1.en.html. (Accessed on 04/19/2017).