Project 1: Getting Acquainted

Courtney Bonn, Isaac Chan

Group #39

Abstract

In the first assignment for the term, we are tasked with making sure our tools work for the term. First, we work through getting the kernel up and running on the os2 server, using the provided files. Once we were successful running the kernel, we build a new kernel and ensure that boots the VM as well. After we successfully built a new kernel, we moved on finding a solution to the producer-consumer problem.

I. Log of Commands

1) cd /scratch/fall2017
2) mkdir 39
3) cd /scratch/fall2017/39
4) git clone git://git.yoctoproject.org/linux-yocto-3.19
5) cd linux-yocto-3.19
6) git status - to confirm we are on tag v3.19.2
7) cd ..
8) source /scratch/files/environment-setup-i586-poky-linux.csh
9) qemu-system-i386 -gdb tcp::5539 -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”
10) gdb (in new terminal tab)
11) (gdb) target remote: 5539
12) (gdb) c
13) root (in VM)
14) cp /scratch/files/config-3.19.2-yocto-qemu /scratch/fall2017/39/linux-yocto-3.19/.config
15) make -j4 all
16) qemu-system-i386 -gdb tcp::5539 -S -nographic -kernel linux-yocto-3.19/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”
17) gdb (in new terminal tab)
18) (gdb) target remote: 5539
19) (gdb) c
20) root (in VM)

II. Explanation of Qemu Flags

To learn what the qemu flags represented, we researched the linux man page [**?**].

- -gdb
  This enables the debug mode.
- tcp::5539
  This specifies the port.
- -S
  This tells the CPU to not start right at startup.
- -nographic
  This disables graphics which makes qemu only display on the command line.
- -kernel
  This is where the kernel file location is defined.

- -drive file=¡¿,if=virtio

  This is where the file that will be used for the virtual disk is defined.
- -enable-kvm

  This enables KVM virtualization and is the reason the VM boots so quickly.
- -net none

  This says there should be no network devices configured.
- -usb

  This enables the USB driver.
- -localtime

  Set the CPU at local time.
- –no-reboot

  Don't reboot qemu, just exit.
- –append "root=/dev/vda rw console=ttyS0 debug"

  This tells qemu to launch in debug mode.

III. Concurrency Write Up

1) What do you think the main point of this assignment is? The main point of the assignment is learn the basics of sharing a resource between multiple processes. The producer-consumer problem is a common problem in parallel processing.
2) How did you personally approach the problem? Design decisions, algorithm, etc. Essentially the problem is having multiple processes trying to write to one shared resource (the buffer). They can't alter the buffer at the same time so there must exist locks (mutexes) when the buffer is being used. The consumer algorithm was as follows: Check for empty buffer, lock mutex, consume buffer item, unlock mutex, increment the semaphore spaces. The producer algorithm was as follows: Check for full buffer, lock mutex, create item, add item to buffer, unlock mutex, increment the semaphore items.
3) How did you ensure your solution was correct? Testing details, for instance. There are three main tests for correct producer-consumer solution implementation. Producer and consumer don't alter the buffer at the same time. The producer won't try to add an event if the buffer is full. The consumer won't try to consume an event if the buffer is empty.

   We tested our solution on the os server and outputted print statements on the state of the mutex. This demonstrated that it was unlocking and locking correctly so the producer and consumer weren't acting on the buffer at the same time. Then we were able to track the state of the buffer and saw that the consumer wasn't acting when it was empty and the producer wasn't acting when it was full.
4) What did you learn? We learned the basics of a kernel and running it in a VM. In the concurrency exercise, we learned a lot about semaphores and mutexes and working with p_threads. We also learned how to embed ASM into C and got more practice debugging with GDB.

IV. Version Control Log

V. Work Log

Date	Time	Person	Event
October 5, 2017	4:00pm	Isaac	Set up shared directory
	4:30pm	Isaac	Run acl_open script to share
	5:00pm	Isaac	Unsucessfully try to start the qemu VM
	5:30pm	Isaac	Sucessfully build the new kernel
October 7, 2017	7:10pm	Courtney	Set up LaTex template
	7:30pm	Courtney	Set up Git repository
	7:55pm	Courtney	Finished fixing issue with Github
	8:30pm	Isaac	Start the concurrency assignment
	8:55pm	Courtney	Tried setting up Overleaf with Github, unable to figure it out rig
	9:01pm	Courtney	Tries running qemu command, gets error that says ”qemu command
	9:39pm	Courtney	Resourced configuration file, successfully runs qemu comma
	9:42pm	Courtney	Opens new terminal and connects to gdb and remote port
	9:45pm	Courtney	Successfully boots VM in qemu
	10:09pm	Courtney	Booted VM using new kernel file
	10:13pm	Courtney	Began adding command log to write up
	10:30pm	Isaac	Set up assignment shell and add todo notes
	11:00pm	Courtney	Added makefile and made sure it correctly build the tex fil
October 8, 2017	1:00pm	Courtney	Fixed the folders on Github to pull from Overleaf correctl
	6:45pm	Courtney	Began researching the qemu flags and adding explanations to the
	7:30pm	Courtney	Started compiling the work log
	8:00pm	Isaac	Continue work on concurrency assignment
	8:10pm	Courtney	Began reading 4.1 of Look Book of Semaphores
	9:00pm	Isaac	Add random number generation
	9:30pm	Isaac	Add threads and mutexes
October 9, 2017	5:30pm	Courtney	Continued researching P-C problem to try and finish the co
	7:00pm	Isaac	Write the rest of the concurrency assignment
	8:00pm	Isaac	First version done, but seg fault
	8:00pm	Courtney	Began debugging Seg Fault error on Concurrency exercise
	8:31pm	Courtney	Fixed Seg fault; Program compiles with no errors or warnings and runs without a Seg fault,
	10:00pm	Courtney	After debugging for an hour and half, finally found the line that was the source of t
	10:10pm	Courtney	Added counter to buffer and now code is entering the producer and cons
	10:40pm	Courtney	Verified everything is working correctly
	10:50pm	Isaac	Began working on my work log and concurrency writeup
	11:00pm	Courtney	Finished up my work log and wrote abstract
	11:00pm	Isaac	Edited the makefile to include the C file
	11:13pm	Courtney	Added Github version control log