Homework 1 Report
CS444 Fall17

Zach Lerew, Rohan Barve
Group 26

October 9, 2017

Abstract

Homework 1 involves sourcing the yocto kernel, building it, and running it with qemu. The second task is to solve a producer/consumer concurrency problem in C.

Command line qemu parameters

All command descriptions learned from linux man pages

- -gdb
  Wait for gdb connection on device dev. Typical connections will likely be TCP-based
- -S
  Do not start CPU at startup
- -nographic
  Normally, if QEMU is compiled with graphical window support, it displays output such as guest graphics, guest console, and the QEMU monitor in a window. With this option, you can totally disable graphical output so that QEMU is a simple command line application. The emulated serial port is redirected on the console and muxed with the monitor (unless redirected elsewhere explicitly). Therefore, you can still use QEMU to debug a Linux kernel with a serial console. Use C-a h for help on switching between the console and monitor
- -kernel
  The kernel image. Can either be a Linux kernel or in multiboot format.
- -drive
  Define a new drive. This includes creating a block driver node (the backend) as well as a guest device, and is mostly a shortcut for defining the corresponding -blockdev and -device options
- -enable-kvm
  Enable KVM full virtualization support
- -net
  Indicate that no network devices should be configured. It is used to override the default configuration
- -usb
  Enable the USB driver (if it is not used by default yet)
- -localtime
  Sets the time to the current UTC local time
- –no-reboot
  Exit instead of rebooting
- –append
  Pass commands to the parameters of the kernel when it is started

Kernel build,test,VM Qemu setup

- mkdir /scratch/fall2017/26.
- cd /scratch/fall2017/26.
- git clone git://git.yoctoproject.org/linux-yocto-3.19.
- cd linux-yocto-3.19.
- git checkout v3.19.2; cd ../
- cp /scratch/files/environment-setup-i586-poky-linux* ./
- cp /scratch/files/bzImage-qemux86.bin ./
- cp /scratch/files/core-image-lsb-sdk-qemux86.ext4 ./
- source environment-setup-i586-poky-linux

- cd linux-yocto-3.19.
- make clean.
- cp /scratch/files/config-3.19.2-yocto-standard ./.config.
- makemenuconfig, adjust general setting Local version to needed group no.
- Save the file as .config and exited menuconfig.
- Compiled the kernel using: make -j4 bzImage. Used 4 threads.
- qemu-system-i386 -gdb tcp::5526 -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"
- Open new terminal session and launch gdb.
- Within gdb connect to specified port: (gdb) target remote:5526.
- This step should result in the kernel booting and a login screen qemux86 appearing.
- The login is root and there is no password so I simply hit Enter.

Concurrency assignment

Purpose

The assignment states that the purpose of concurrency assignments is to hone your skills in thinking in parallel. Parallel programming is a large independent topic worthy of study. When it comes to operating systems, synchronous work is less common and less useful than parallel async work. This assignment helps us practice those skills.

Approach

The problem was fairly easy to solve when you think in terms of objects and analogies.
There are items in a buffer, which must be created by producers, and consumed by consumers.
The buffer can be thought of as a cookie jar with a small opening, only one hand can get in at a time.
Once an item is taken out of the jar, another hand can get into the jar to do its job.
Items can be created and consumed outside of the jar (in parallel), and then fight for a chance to access it when they need to.
Out of this analogy easily comes the pieces needed to solve this problem:

- An item with some data.
- A buffer to store those items.
- A consumer who locks the mutex, checks the buffer for an item, takes one if it can, and then unlocks the mutex.
- A producer who locks the mutex, checks if the buffer is full, adds a new item if it can, and then unlocks the mutex.
- And lastly a main function to spawn threads and initialize data.

Testing

With the above approach on paper, a loosely TDD approach was taken.
This problem was small enough however, that a manual test could be used for each step, rather than a traditional failing unit test.
Final results were independently verified by both team members.

Learned

Every time I work with C, I am reminded of its power, as well as its drawbacks. This is a fairly simple problem to solve, which makes C an overly complicated tool to solve it with.
In a professional work environment, the tools that can be used are typically limited to the code base owned by the company.
Regardless of the problem and the tools that exist, if the code base is in C#, the solution should be too (with occasional exceptions).
I understand that the purpose of the class and assignment is low level kernel programming in C (the language of this ćode base). However, why use a 700 bhp v8 in a 4 door sedan that you drive to the grocery store in bumper-to-bumper traffic?

Version control log

Date	Author	Description
2017-09-30	Zach Lerew	Initial directory and hw1 start
2017-09-30	Zach Lerew	Named makefile correctly, updated rules
2017-09-30	Zach Lerew	Consumer threads created and run correctly
2017-09-30	Zach Lerew	Consumer threads block until they finish consuming
2017-09-30	Zach Lerew	Consumer threads wait for buffer items, producer work func made
2017-09-30	Zach Lerew	Consumers lock and unlock mutex to allow producers a chance to produce
2017-09-30	Zach Lerew	Implemented random number generation using asm() and Mersenne Twister
2017-10-01	Zach Lerew	Supporting CS444 syntax preferences
2017-10-01	Zach Lerew	Added lerewz portion of writeup and latex document base
2017-10-09	Rohan Barve	Adding modified writeup.tex
2017-10-09	Rohan Barve	Adding instructions to run kernel for future use
2017-10-09	Rohan Barve	Added related misc files
2017-10-09	Zach Lerew	Added git log and abstract to write up
2017-10-09	Zach Lerew	Removed unnecessary qemu parameters

Work log

This assignment has three distinct sections:

- Kernel building and qemu emulation
- Consumer & Producer concurrency problem
- This document

The work was split between us as such -
Zach: Concurrency problem, write-up base, makefile, git log
Rohan: Kernel and VM Qemu setup, configuration and test , concurrency problem review, write-up body