Project 1

CS444 - Operating Systems II

Fall 2017

Instructor: D. Kevin McGrath

Xiaoli Sun, Jaydeep Hemant Rotithor

Abstract

This is the first CS444 assignment. There are two parts to this assignment. The first part is to build a kernel and then let the kernel boot on the virtual machine. We do this because we need to learn how to build a kernel and set up a virtual machine. The second part is to do a concurrency exercise using C programming. In this paper, we will document how we build a kernel on the VM in each step, and describe how to design a concurrency program in C.

Oct 3, 2017

1 Build kernel and run VM

Step 1: mkdir 35

The first step is to create a folder under /scratch/fall2017.

Step 2: git clone git://git.yoctoproject.org/linux-yocto-3.19

The second step is to clone the kernel from github to the folder created in step 1.

Step 3: git checkout tags/v3.19.2

The third step is to switch tags to v3.19.2 under /scratch/fall2017/35/linux-yocto-3.19

Step 4: cp config-3.19.2-yocto-standard ./linux-yocto-3.19/.config

The fourth step is cd /scratch/files, then copy config-3.19.2-yocto-standard to the directory of the root of linux tree: ./linux-yocto-3.19/.config. After that, copying bzImage-qemux86.bin and core-image-lsb-sdk-qumex86.ext4 to /scratch/fall2017/35/linux-yocto-3.19

Step 5: source /scratch/files/environment-setup-i586-poky-linux.csh

The fifth step is to cd to linux-yocto-3.19 directory, and source the csh file.

Step 6: make -j4 all

We can build the kernel in the sixth step by inputting make -j4 all command. 4 threads takes about 5 minutes to finish.

Step 7: qemu-system-i386 -gdb tcp::5535 -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"

In the seventh step, we changed ???? to 5535 and then we are ready to boot it on the VM.

Step 8: gdb

The eighth step is to connect gbd to a remote target at the specified port(5535). First we open a new PUTTY window, connect to the os2 server and change the directory to /scratch/fall2017/35/linux-yocto-3.19. After that, simply entering gdb.

Step 9: target remote :5535

The ninth step is to connect gbd to a remote target at the specified port(5535). After that, entering "file bzImage-qemux86.bin".

Step 10: continue

The tenth step is to continue the process and will cause the OS to boot.

Step 11: root

In the eleventh step, we change to the first PUTTY window and find that the kernel is already booted on the VM. Then we login in as root and no password is required.

Step 12: shutdown -h now

In the twelveth step, we can use the command above to shut down VM.

Concurrency solution

In this problem, we should first understand that there are two processes, consumer and producer. There's also a buffer shared by the two processes. The producer will produce a number and put it into the buffer. Meanwhile, the consumer will delete this number from buffer. When the buffer is full, block the producer until consumer deletes a number from buffer. When the buffer is empty, block the consumer until producer adds an number to buffer. The problems that we met when solving this question are when and where the mutex should be blocked, how to correctly block consumer and producer, etc.

2 Flags explanation

-gdb tcp::5535
It tells qemu to wait for gdb connection from port 5535.

-s
Don't start CPU when boot.

-nographic
Disable GUIs and use command line.

-kernel
Use bzimage as a boot img.

-drive file=core-image-lsb-sdk-qemux86.ext4
Let qemu use core-image-lsb-sdk-qemux86.ext4 as disk image.

-enable-kvm
Enable the KVM for qemu.

-net none
kernel won't use any network devices.

-usb
enable usb devices.

-localtime
set the time as local time.

-no reboot
no roob when exit kernel

–append "root=/dev/vda rw console=ttyS0 debug"
use "root=/dev/vda rw console=ttyS0 debug" to run the command line in the linux kernel.

3 Concurrency questions

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

In my opinion, the main purpose of this assignment is to consolidate my understand of concurrency. Concurrency is a very important concept in operating systems. Modern operating systems allow multiple processes run at the same and access same resources at the same time as well. This assignment also let me practice how to deal with the situation like resources are empty or full when run multiple process access the same resources.

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

First of all, I carefully read the question and found that one structure was required and the structure should contains two integers, a number and a period waiting time. I also created three threads by using pthread, two for consumer and producer, the rest for mutex. Two processes will have a shared resources which could hold 32 items. I created a int array and define the array size as 32. Afterwards, I created two functions: produce and consume. In produce function, I first let producer sleep for a random number of seconds, the time is generated by genrand_int32() function. The genrand()_int32() function is defined at "mt19937ar.c" file and I included it in my C code. Then the function will check if the buffer size if full or not. During checking the buffer, the buffer is locked by using mutex. If the buffer is full, the function will wait for 2 to 9 seconds to add a new number to the buffer until consumer send the signal to producer to inform the buffer is no longer full . Then assign the random generated new number and waiting period to structure. Finally, print the result. The consumer has the same approach as the producer.

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

I tested my code multiple time by changing buffer size. When buffer size is equal to 0, consumer will not delete number but wait for seconds until producer send a signal to it. When buffer size is equal to 32, producer will stop generating new number until it receives signal from consumer. I also testing waiting period to see if consumer and producer actually sleep for the time defined in the code.

4. What did you learn?

In this assignment, I learned how to correctly run multiple processes simultaneously and access the same resource at the same time. I also learned how to write make file for pdf and how to use Latex. Finally, I learned that CS444 is not all based on writing codes, it's more focus on important concepts of operating systems.

4 Version Control

Revision	Date	Author(s)	Description
7909a21	10.5.17	Xiaoli Sun	Created central repository.
3ad1310	10.5.17	Xiaoli Sun	Upload concurrency C code.
8052f07	10.6.17	Xiaoli Sun	Upload all files for concurrency.
5851fc3	10.6.17	Xiaoli Sun	Add pdf file and source file for write up.
b79fb75	10.7.17	Xiaoli Sun	Update source file and pdf file for write up.
fae9967	10.7.17	Xiaoli Sun	Upload final version of write up.

5 Work log

Date	Summary
Friday, 9/29	Looked over project requirements and began planning.
Saturday, 9/30	
Sunday, 10/1	
Monday, 10/2	Created the group repository under scratch/fall2017 and cloned the linux-yocto-3.19 repository.
Tuesday, 10/3	Successfully ran qemu in debug mode with GDB.
Wednesday, 10/4	Began write-up.
Thursday, 10/5	Updated the .config file, compiled the kernel, and successfully got it to run.
Friday, 10/6	Finished write-up and prepare to submit the assignment.
Saturday, 10/7	Fixing LaTeX Error when compiling latex source code.