Writeup 1

CS 444
Spring 2017

Abstract

This writeup is the writeup for Project 1 for CS444 Operating Systems 2. The write-up includes the Linux Yocto kernal on os-class.engr.oregonstate.edu and the solution of Producer-Consumer in C.

Contents

1 Concurrency writeup

The producer-consumer problem is a well studied problem. Algorithms were simply found from a search and used the POSIX threads and lpthread build from the linux command line. After producing and creating thread functions were created, these were used to pass items back and forth before joining. In order to prevent simultaneous access, mutexes were used, along with random waiting period, where threads would be blocked from accessing the array of items if another thread was accessing at the time.

2 Log of Commands to Build Yocto Kernel and Load Qemu

1. Open two terminal windows. In terminal #1, do steps 2-12. In terminal #2 do step
2. Log on to os-class : pengc@os-class.oregonstate.edu(Please use your own user nam
3. Call cd scratch/spring2017/10-01 . If you don't have this folder(10-01), call m
4. Call git clone git://git.yoctoproject.org/linux-yocto-3.14 to download the proje
5. To switch to the correct tag, call cd linux-yocto-3.14, and then git checkout v3
6. Before we build our kernel or run qemu, we should configure the environment, so
7. Follow 8-12 steps, make a kernel instance for your group.
8. Run cp /scratch/spring2017/files/config-3.14.26-yocto-qemu .config
9. Run make menuconfig and you will get a window
10. In the widow do the following: press / and type in LOCALVERSION, press enter.
11. Hit 1, press enter and then edit the value to be -10-01-hw1 (-10-01-hw1 for grou
12. Run make -j4 all, your kernel instance will be built with 4 threads
13. Run cd .. and then run gdb. Stop here for now.
14. In terminal #2, do step 6.
15. To make a copy for the starting kernel and the drive file located in /scratch/sp
16. Call cp /scratch/spring2017/files/bzImage-qemux86.bin . (This . is an operand)
17. Call /scratch/spring2017/files/core-image-lsb-sdk-qemux86.ext3 . (This . is an
18. Try run the starting kernel : Call qemu-system-i386 -gdb tcp::5601 -S -nographic
19. Since in step 18, we run qemu in debug mode with the CPU halted. We need to use
20. In terminal #1, it is now in gdb. Run target remote :5601 to connect the qemu.
21. Run continue. Then you will see the change in terminal #2.
22. If you succeed in running qemu, you will be asked to login. Type root and enter
23. Use reboot to reboot the VM.
24. Try run the kernel instance we created in steps 8-12. The kernel instance we bu
25. Run qemu-system-i386 -gdb tcp::5601 -S -nographic -kernel linux-yocto-3.14/arch/
-drive file=core-image-lsb-sdk-qemux86.ext3,if=virtio -enable-kvm -net none -usb -localt
26. Do steps 20-22 again. You should find the difference in kernel names. The name s
27. reboot the vm and use q to quit gdb.

3 Flags in the listed Qemu command line

The listed Qemu command line is:

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".

The following list describes each flag:

- *-gdb* instructs qemu to wait for input from the subsequent flag.
- *tcp::5628* gives qemu the device to listen on for gdb input.
- *-S* tells qemu to not start the CPU at startup.
- *-nographic* disables graphical output so qemu appears as a command line application only.
- *-kernel* followed by a file specifies the kernel to be used, in this case the bzImage we created.
- *-drive* defines a new drive; the following arguemnts specify the disk image file (file=core-image-lsb-sdk-qemux86.ext3) and the interface option (if=virtio).
- *-enable-kvm* enables KVM virtualization, a very complicated function that assists in emulating hardware.
- *-net* interfaces with network options; assumedly as it is followed by "none" the VM does not have networking support.
- *-usb* enables the USB driver.
- *-localtime* enables the machine to start at the current local time.
- *–no-reboot* configures the system to shutoff instead of reboot.
- *–append* tells qemu to use the following argument ("root=dev/vdarw console=tty50 debug") as the kernel command line.

4 Concurrency

The following subsections answer the four questions as outlined on the Project 1 page on Kevin McGrath's course website.

4.1 Main point of assignment

The main points of the assignment were to remind us and refresh our usage of the pthreads in C which we did one (maybe 2?) assignments in CS344 which was probably a year removed from most of us in the class. In addition, it showed us a practical way the multiparallel processing is useful in a possible real life scenario of producer-consumer, of course in which real life is much more complicated. Third, if some of us had not programmed in C in a while (my last couple CS classes have been in python and java), it was a good refresher for that as well.

4.2 Personal approach to problem

My approach comes from some research with the producer-consumer problem. The pthreads documentation is pretty easy to find so functions such as create, join, mutex, etc. were all used in thsi problem. Because the easiest way to make the buffer was a array of structs, it made more sense to use static threads in which the buffer has 2 data fields, a random val and a random waiting time. Having a thread for producer and consumer thread made it quite easy; the producer would lock the thread using a mutex then produce and item. After unlocking, the consumer would lock it again and consume the buffer, and then unlock again. Once this runs x number of times, an internal counter would make the program return. A producer could always grab the mutex and add items, depending on the order of the thread creation but only up to how full the buffer can get.

4.3 Verification of solution

The important parts that needed to be verified were when the counter applied and which thread grabbed control of the mutex. To verify the solution, print statements were used. The insertion from the producer at the index location should be the same as the consumer retrieving the random value from the same index position and the value should be the same also. Print statements were added that would print out the values of the producer and consumer, as well as their sleep times. The index goes from 1-32 so the index shouldn't be a problem. Using pthread conditionals, it is impossible to have both threads acessing the mutex at the same time and it is easy to tell which one because the print statement will state which thread it is. I experimented on the the code with arguments of 1-5 and the output showed that it was called the proper number of times.

4.4 What I learned

Austin reviewed knowledge of concurrency, and with the synchro2 problem discussed in the recitation, is starting to get into the parallel mindset. Using mutexes again helped this process to remember to not allow multiple threads to access the same data. The assignment was a good warmup for any future problems where a good solution would be a parallel one.

5 Version control log

History of versions

5.1 Git Logs

2eb5240 was Austin, 2 minutes ago, message: working on verification
2d4a72b was Austin, 9 hours ago, message: 4/21 some more changes outside verification
c40e630 was Austin, 9 hours ago, message: more tex edited 8badf40 was Austin, 8 minutes ago, message: pdf test creation 597d814 was Austin, 43 minutes ago, message: Tex edit
304e7e8 was IStallcup, 4 hours ago, message: small changes to concurrency
5336067 was IStallcup, 5 hours ago, message: Merge branch 'master' of https://github.com/Boeinco/CS444
6 fa3edf was IStallcup, 5 hours ago, message: added qemu flag explanation
5ffd876 was Isaac Stallcup, 5 hours ago, message: Create .gitignore
66d601f was Austin, 12 hours ago, message: Writeup work
53de57c was Austin, 12 hours ago, message: fixed command arg c fb1942 was Austin, 32 hours ago, message: writeup updated
401f05e was Austin, 32 hours ago, message: writeup updated
8425628 was IStallcup, 33 hours ago, message: Added some more stuff
cf27ae5 was IStallcup, 33 hours ago, message: added some writeup formatting
6e073c6 was Austin, 34 hours ago, message: Template tex file done
dae7e2e was Austin, 35 hours ago, message: concurrency mostly done
4d7f1cc was Austin, 2 days ago, message: temp files
3e7be70 was Austin, 2 days ago, message: some files in
599f66e was Austin, 2 days ago, message: started concurrency
c6a4f0c was Austin, 2 days ago, message: Testing directories
8e404a9 was Austin Nguyen, 3 days ago, message: first commit

Version	Date	Author(s)	Changes

6 Work log

6.1 April 18-17

Austin created git project and started working on concurrency. Used latex template to set up the latex document and git commits under username Boeinco.

6.2 April 19-17

Austin started working on the new project requirement of the command line argument after creating the main project frame. After creating mutexes and having them sleep for the specified random amount of time, work was done on translating it to the producer/consumer problem.

6.3 April 20-17

Austin finished up the producer/consumer problem and added the qemu instructions. Isaac did research on the flags during the linux commands and also helped format the latex file. Some latex work was done.

6.4 April 21-17

Austin did the section of version control and other parts of the document.