Project 1

Fall 2017 CS544 Operating System

Weijie Mo

October 9th 2017

Abstract: This report is a description of what I've done in this project, including problem and solutions. There are also other essential parts like a log of commands, explanations about qemu's flag, answers of listed questions, work log.

I. POMMAND LOG

- *git clone git://git.yoctoproject.org/linux-yocto-3.19* That creates a directory named *linux-yocto-3.19*, initializes a .git directory inside it, pulls down all the data for that repository.
- *git checkout v3.19.2* Switch branches or restore working tree files.
- *cp /scratch/files/config-3.19.2-yocto-qemu ./.config* Copy /scratch/files/config-3.19.2-yocto-qemu to .config.
- *source environment-setup-i586-poky-linux* Setting the environment for building the kernel and running qemu.
- *make -j4 all* Uses 4 threads to build the kernel.
- *qemu-system-i386 -gdb tcp::5556 -S -nographic -kernel 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"* Boot the kernel on the VM. The group port is 5556, and we use our bzImage kernel that is in the /arch/x86/boot directory.
- *gdb* gdb begins to debug the kernel.
- *target remote :5556* set the gdb target to the port 5556.
- *continue* continue the running.

II. EXPLAIN OF QEMU COMMAND-LINE

- *-gdb tcp::5556* Wait gdb to target the port 5556 to continue the running of the QEMU.
- *-S* Do not start the CPU at the start point.
- *-nographic* Normally, QEMU uses SDL to display the VGA output. With this option, I can totally disable graphical output so that QEMU is a simple command line application. The emulated serial port is redirected on the console. Therefore, I can still use QEMU to debug a Linux kernel with a serial console.
- *-kernel bzImage* Use bzImage as kernel image.
- *if=virtio -enable-kvm* Enable KVM full virtualization support. This option is only available if KVM support is enabled when compiling.
- *-net none* presents that no network devices should be configured.
- *-usb* Enable the USB driver
- *-localtime* For correcting date in Windows.
- *–no-reboot* Exit instead of rebooting.
- *–append* Use cmdline as kernel command line
- *-drive* Valid options I have used is file=file This option defines which disk image (see section Disk Images) to use with this drive. If the filename contains comma, you must double it (for instance, "file=my,file" to use file "my,file".

III. SOLUTION OF CURRENCY1

In this assignment, I use three semaphore to control the mutual exclusive access to the buffer. First, a mutex semaphore is used to make sure only one people no matter it is the producer or the consumer can access the buffer.The mutex is initialized to 1. Whenever one person access the buffer, it will decrease mutex semaphore and lock it. This means that no other people can access the buffer anymore. Any other people have to wait this person release the mutex which will increase its value.

After the processing of the buffer such as printing out the value of the item, the person who is accessing the buffer will unlock it.

Second, I use one emptySem semaphore to ensure that no consumer can get any items when the buffer is empty. This semaphore is initialized to 0, and be increased by the producer whenever the producer access the buffer and create one item. This means the emptySem semaphore will be increased. Whenever one consumer get one item, this semaphore will be decreased. If the buffer is empty, then this semaphore will become negative or zero, and all consumers who finished consuming will start to wait until any producer add one item.

At last, a fullSem semaphore is created to control the buffer size. This semaphore is initialized to 32 based on the requirement of the assignment. Then any producer who add one item will decrease its value by sending the signal. If this semaphore is negative, which means that all the spaces are used to store items. Then no more items can be added. This semaphore will be increased whenever one consumer get one item. Then all the waiting producer can get space to store item again.

IV. Work log

1) *Wed Oct 4 2017* I start to figure out git operation questions,including some commands needed to understand and keep them in my mind.
2) *Fri Oct 6 2017* Get to read the first two chapters of the book and learned some ideas about synchronization from this book.
3) *Sat Oct 7 2017* Continue to read the book, besides I also get more familiar with the LaTex,like how to use it write a report and essay,and begin to write my own report
4) *Sun Oct 8 2017* Solved consumer and producer problem and begin to work logs in Latex format.
5) *Mon Oct 9 2017* Completed the Latex format and debug all project, to ensure there is no problem in the assignment.

V. Git log

Author	Date	Commit message
Weijie Mo	Mon Oct 9 2017	remove the readme
Weijie Mo	Mon Oct 9 2017	edit the report
Weijie Mo	Sun Oct 8 2017	start the writing
Weijie Mo	Sun Oct 8 2017	project added
Weijie Mo	Sun Oct 8 2017	dircreated
Weijie Mo	Sun Oct 8 2017	readme added

VI. Questions

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

There are so many new tools, concepts, theories, commands during the process of doing this assignment, I think such things are essential for the next periods study, that is the main point of this assignment, which also help me get more familiar with these new knowledge.

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

First, I leaned about the semaphore and synchronization patterns from *the little book of semaphore*. From this book, I leant how to analyze synchronization problems with considering exclusive access. Then I start to read the materials in class website about gdb, gcc, makefile, mobXterm, cgwin, git, bash, etc. Since I don't have any developing experience in Linux, this process takes me long time.

After all these preparation, I start to code the concurrency exercise in cygwin. With the help of the demo codes about generating random value and create threads on the web page, this process is not that hard. The implementation of the algorithm of the producer and consumer problem is based on the pseudomonades on the book.

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

I just read the book and it inspired me with the example of who eat lunch first, to ensure the final value is as expected as before, I set two threads , first is a producer thread, second is consumer thread, I will not create the second thread until I ensure the first one could run successfully. In the end ,I created both of these threads.

D. What did you learn?

I have mastered plenty of new knowledge in this assignment. I learned how to install a VM , what producer-consumer problem is, build my new kernel as required, I acquainted many new tools like gdb, gcc, makefile, mobXterm, cgwin, git, bash, etc. During the period of working on this project, I must keep a balance of mastering new knowledge and doing the project, because both cost so much time and energy, however the book and my friends helped me a lot, and finally I made it.

REFERENCES

[1] The little book of semaphores[M] *Downey A.* [*Allen B. Downey, 2016.*].