Project One Writeup

Spring 2017

Shuai Peng (pengs), Anya Lehman (lehmana), Andrew Bowers (bowerand),

April 21, 2017

Abstract

This is our write up for the project one *Getting Acquainted*. This project included setting up our environment and an exercise on concurrency. Listed below is the process we took to set out the environment as well as the process we took to complete the concurrency exercise. Note: All text in red is code.

1 Command Log

1) To start we logged on to the os-class ssh username@os-class.oregonstate.edu
2) Then we used cd to get to the correct folder in scratch/spring2017 cd /scratch/spring2017
3) Then we made a group folder for us all to work in mkdir 11-04
4) Next we struggled to make said folder accessible to all of our gorup members by changing the permitions on it so not just the group member that cr eated the directory could work in it chmod 777 11-04
5) Then we called git clone to download the project from the GitHub account and we checked to make sure we got all the correct files git clone git://git.yoctoproject.org/linux-yocto-3.14
6) Then we switched to the tag we needed by using cd again and going into the directory that was cloned into our folder cd linux-yocto-3.14
7) Following this we checked out the v3.14.26 git checkout v3.14.26
8) Next came configuring the environment which we did by calling source /scratch/opt/environment-setup-i586-poky-linux
9) Then we made a kernel instance for our group
10) Then we coppied in the files that lets us configure cp /scratch/spring2017/files/config-3.14.26-yocto-qemu .config
11) Then we ran make menuconfig
12) A window popped up
13) In this window we pressed / and typed LOCALVERSION
14) Next we pressed 1 and edited the value to be -11-04-hw1 to make that the name of the kernel
15) Then we built our kernel with four threads by running -j4
16) Then we ran cd .. followed by gdb
17) Our next step was to move onto a different laptop and called source /scratch/opt/environment-setup-i586-poky-linux again
18) Then we made a copy for the starting kernel and the drive file located in the scratch directory by calling /scratch/spring2017/files image-lsb-sdk-qemux86.ext3
19) Then we tried running the starting kernel qemu-system-i386 -gdb tcp::5604 -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"
20) Since we previously ran the qemu in debug mode, we used gdb to control it so, back on the origonal computer, we connected the qemu by running target remote :5604
21) Then we rebooted the VM reboot
22) Then we tried running the kernel instance we had created linux-yocto-3.14/arch/x86/boot/
23) Then we ran qemu-system-i386 -gdb tcp::5601 -S -nographic -kernel linux-yocto-3.14/arch/x86/boot/bzImage -drive file=core-image-lsb-sdk-q emux86.ext3,if=virtio -enable-kvm -net none -usb -localtime –no-reboot –append "root=/dev/vda rw console=ttyS0 debug"
24) Finally we rebooted the vm and used q to quit

2 Write Up of Concurrency Solution

We noticed that the assignment required functionality for a consumer, producer, and random number generator operations. The best solution we thought of was to create functions for each of these operations, then use main to generate our threads and handle the operations of the program. Our main function intitialzes our values, creates the threads, then loops through calling the different functions as necessary. Once the producer function is called, it calls the random generator function to create the numbers to be stored in the struct, which are stored in the buffer once the struct is created. We used mutex locking to protect data from other threads. Theproducer then waits it's wait period, which is also randomly generated. In a similar fashion, when the consumer fucntion is called it will pull a struct from the buffer, wait for the given amount of time, given by the second number in the struct, then print out the first number. As noted earlier, main loops through creating threads and calling the functions.

3 Explaition of the Flags

1) -gdb makes the device wait for a connection to the gdb debug tool.
2) -nongraphic means make it a non graphical interface that works like the command line.
3) -kernal means to use the following kernel.
4) -drive means to use the following driver.
5) -enablekvm enables KVM in full virtualization support.
6) -net means to use the follwing network.
7) -usb enables the USB driver.
8) -localtime its setting the time for the local machine.
9) –no-reboot exits instead of rebooting.
10) –append uses the command line as the kernel command line.
11) -s changes it to strict mode which makes it so that it fails on different image size or sector allocation.

4 Questions Regarding Concurrency

1) *What do you think the main point of this assignment is* The main point of this assignment is to get us to learn more about conncerency and to build up our ability to understand and think in parallel. Parallel computing is the simultaneous use of multiple resources to solve a computational problem. This is important because compared to serial computing, parallel computing is much better for tasks such as modeling, simulating, and working on more complex, real world problems. The conceptual problem of this assingmnt is not as challenging as understuanding how to implement it. The secondary point of this assignment is to get us familiar with the techniques we need for the rest of this class.
2) *How did you personally approach the problem? Design decisions, algorithm, etc.* To start this problem, we first worked out the pseudo code with pen and paper. We designed it so that the Consumer is a function and the Producer is a function. Then we also created a function that generated different random numbers based on cases**[1]** **[3]** (case for the 3-7 second waiting period, the 2-9 second waiting period, and the random number that the consumer prints out). We went about this by using a struct to hold the values that the Producer creates and the Consumer are taking. The first number in each struct will be the randomly gemerated number for the consumer to print and the second

will be the time the consumer should wait. This struct is generated in our main function. We also created a function that checked to see if RdRand works on the current system.**[2]**

3) *How did you ensure your solution was correct? Testing details, for instance.* To ensure that our solution was correct we used printf to print out all the step by step details that are not normally shown so that we knew exactly what was going on behind the scene.
4) *What did you learn?* This assignment taught us how to properly create a program that uses parallel programming. We re-learned a lot of old material about writing in c and utilizing pthreads. We also learned the control you can gain with inline assembly. This project also helped us understand how we work as a group.

5 Version Control Log

File Version	Group Member(s)	What Was Done
V1	Andrew	Created the concerency file to work from
V2	Andrew	Developed the functions
V2	Shuai	Implemented the functions
V3	Andrew	Developed the structs and the consumer and producer
v4	Shaui	Adapted the mt19937
v5	Shaui	Debugged the file
v6	Anya	Created the rdrand function
v7	All	Compiled it all together

6 Work Log

Date	Group Member(s)	Start Time	End Time	Total Time Worked	Accomplished
April 11th	All	11:00am	12:00pm	1 hour	Met our group members and exchanged contact information. Set up our working environment including creating our kernel and making the group folder to work in. Made sure all members of the group could access the group folder and the group GitHub account.
April 18th	All	11:00am	12:00pm	1 hour	Set up the folder to contain the work we will be doing in this project. Started on the writeup by setting it up and completing the write up for the log of commands. Got the github to work. We also read through the concerency assignment.
April 19th	All	2:30pm	8:30pm	5 hours	Began the concerency assignment. Created the psudeo code for the exersise and discused how we wanted to go about completing the exersise. Created the c file and made our struct and all of our functions. Also worked on the write up to create the work log and begain to asnwer the questions regarding the concerency exercise.
April 20th	All	5:00pm	10:30pm	5.5 hours	Almost finished the concerency assignment. Debugged the project and made it so that the loop went for our desired length. Also made the two random generating stlyes
April 21th	Shuai	12:00pm	1:00pm	1 hour	Fixed the compile error
April 21th	Andrew, Anya	12:00pm	7:45pm	7.75 hours	Finished the concerency assignment. Redid most of the concerency. Edited the makefile.. Also wrapped up the writeup and turned it all in.

7 CITATIONS

REFERENCES

[1] Takuji Nishimura and Makoto Matsumoto. *Mersenne Twister with improved initialization*. MT, 1997-2002.

[2] Blaise Barney. *POSIX Threads Programming*. Lawrence Livermore National Laboratory, 2017.

[3] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual: Basic Architecture'*. Order Number: 325462-060US, Intel technologies, 2016.

[4] Dirk Eddelbuettel. *Catch trl-C in C*. Stack Overflow, 2016.

[5] IBM Knowledge Center. *pthread kill – Send Signal to Thread*. IMB