CS444 Project 1 Group 36

Namtalay Laorattanavech
Dakota Zaengle

Abstract

The purpose of this assignment is to build our first kernel and understand the steps to run it from the OS2 server. In order to refresh parallel programming we also complete a concurrency exercise using pthreads.

Building the Kernel

FIRST PUTTY SESSION

1. cd /scratch/fall2017
2. mkdir 36
3. cd 36
4. git clone git://git.yoctoproject.org/linux-yocto-3.19
5. git checkout v3.19.2
6. cp /scratch/files/bzImage-quemux86.bin bzImage-quemux86.bin
7. cp /scratch/files/core-image-lsb-sdk-quemux86.ext4 core-image-lsb-sdk-quemux86.ext4
8. source /scratch/files/environment-setup-i586-poky-linux.csh
9. make j4 -all
10. qemu-system-i386 -gdb tcp::5536 -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.

START NEW PUTTY SESSION

1. source /scratch/files/environment-setup-i586-poky-linux.csh
2. gdb
3. target remote localhost:5536
4. continue

BACK TO FIRST PUTTY SESSION

1. root
2. uname -r

Qemu Flag Definitions

- qemu-system-i386: Starts Qemu
- -gdb tcp::5536: Waits for a connection from gdb on port 5536 to start
- -S: Pauses and waits for continue command on startup
- -nographic: Causes Qemu to be run through the command line
- -kernel bzImage-qemux86.bin: Tells Qemu what kernel image to use
- -drive file=core-image-lsb-sdk-qemux86.ext4,if=virtio: Defines a new disk drive, virtio means it will use the virtio architecture to set up a virtual drive
- -enable-kvm: Turns on the Kernel-based Virtual Machine enabling full virtualization.
- -net none: Sets network options to none, overriding the default
- -usb: Enables USB to be passed through to qemu
- -localtime: lets the RTC start at the current local time
- no-reboot: Exit instead of rebooting
- append root=/dev/vda rw console=ttyS0 debug.: Use cmdline as kernel command line, gives command line arguments

Concerrency

SOLUTION

The main solution to this concurrency problem is to create producer and consumer in a way that they can tell each other that the items are ready to be consumed or need to be produced. For this, we can use threads functions to allow them to communicate. Also, the buffer will be in a form of a struck since it allows us to store more than one data in one unit. The more detailed steps that we did to solve this problem can be found under question number two.

QUESTIONS

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

The main point of this assignment is to let us review the material from Operating System I class. Since the major part of this assignment is to manipulate threads, it requires us to look back to what we learned in OS I class and apply it to this assignment.

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

First of all, we tried to understand the problem and then came up with a rough algorithm to help us design the program. Then, we looked up how the mt19937ar.c file functions. After we knew roughly what we should do, we did some research by looking at OS I course materials and some quick google search about multi threads programing and ASM in C language. Then we started coding by making a struct for the buffer and main function first, then we carefully built the producer function. Using the producer function as a guide, we successfully coded the consumer function. Then we got back to the main function and filled everything in the way that consumer and producer supposed to interact.

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

Since our program has print statements for every actions consumer and producer do, it was not that hard for us to track the results and test our program. We did a number of tests by using different amount of items. For each test, we checked if the interactions between producer and consumer were correct or not.

4. What did you learn?

We got a chance to refresh our memory about multi threads programing in C. In addition, we also learn something new like how to use ASM in C language and the Mersenne Twister. Last but not least, we also learn how to use LaTex to create this document.

Work Log

10/6/17	Dakota Zaengle	Built Kernel
10/6/17	Namtalay Laorattanavech	Wrote Concurrency
10/8/17	Both	Reviewed both parts of project and started writeup
10/9/17	Both	Finished writeup