CS444: Assignment 1

Spring 2016

Garrett Amidon

April 18, 2016

Abstract

This assignment was about building and developing a kernel envirnoment as well as working on coding in parallel using pthreads. This document includes the steps invovled in initializing the git hub repo, building the kernel, concurrency exercise, defining the qemu flags, version control log, work log and source code.

I. Command Log

1) cd /scratch/spring2016
2) mkdir CS444-007
3) cd CS444-007
4) git clone git://git.yoctoproject.org/linux-yocto-3.14
5) git checkout v3.14.26
6) source /scratch/opt/environment-setup-i586-poky-linux.csh
7) cp /scratch/spring2016/files/bzImage-qemux86.bin
8) cp /scratch/spring2016/files/core-image-lsb-sdk-qemux86.ext3
9) cp /scratch/spring2016/files/config-3.14.26-yocto-qemu.config
10) make -j4 all
11) qemu-system-i386 -gdb tcp::5507 -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”
12) opened new putty
13) gdb
14) target remote:5507
15) continue

II. Qemu Flag Defintions

Found using [1]

A. -gdb tcp::5507

open a gdbserver on TCP port 5507

B. -S

Do not start the cpu on start up

C. -nographic

Disable graphical output so that qemu is a simple command line application

D. -kernel bzImage-qemux86.bin

Use bzImage-qemux86.bin as a kernel image

E. -drive file=core-image-lsb-sdk-qemux86.ext3,if=virtio

Define a new drive with the file set for the disk image if the drive is connected.

F. -enable-kvm

Enable KVM full virtualization support

G. -net none

Indicate that no network devices should be configured

H. -usb

Enable the USB driver

I. -localtime

Specify the RTC to start at current local time

J. –no-reboot

Exit instead of rebooting

K. –append ”root=/dev/vda rw console=ttyS0 debug”

Use root=/dev/vda rw console=ttyS0 debug as command line

III. Concurrency Questions

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

The main point of this assignment I believe was to test my ability to code in parallel using pthreads. The assignment challenged my programming skills and this assignment was also a wakeup call to how much work this class actually will be.

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

To start, I looked up creating ”Hello World” using pthreads because I was not taught how to use a pthread, just normal threads. After familiarizing myself with pthreads on a basic level, I then looked up the pthread library to see the different functionality of a pthread and also researched the problem [2]. From there I read about conditional pthreads and mutex variables.

C. How did you ensure your solution was correct?

To ensure my solution was correct, I tested using 1 as my buffer size max so I could test that the producer would not keep adding to the buffer when it was full and the consumer would not try to remove from the buffer when nothing was there. The first few times I ran this, my program would halt when the buffer reached maximum size. I then narrowed my issue down to the producer function and found that I was not signaling my consumer when there was more data in the buffer.

D. What did you learn?

I learned that I need to start the coding assignments sooner. I had built the kernel last week and figured that the rest of the assignment would not be that bad, I was wrong. I also learned a considerable amount about conditional pthreads and mutex variables.

IV. Version Control Log

Date	Additions	Deletions	Message
4/11/2016	6900	0	Update tex file
4/11/2016	77	79	Finished Concurrent 1
4/11/2016	347	39	Concurrent Progress, need random
4/11/2016	48	0	Adding Concurrent Folder/Progress
4/10/2016	471	0	Clean up/ Week 2
4/10/2016	42	47	Week 1 Summary
4/10/2016	4	0	Merge branch ’master’ of https://github.com/amidongarrett/cs444
4/10/2016	49	0	Adding Weekly Summaries Directory
4/6/2016	4	0	Create README.md

V. Work Log

- Saturday 4/2 - Started working on building the kernel. Worked for about 3 hours trying to figure out what the assignment was asking and trying (failing) multiples times in building the kernel. I finally resorted to actually reading the book (surprise it is helpful) and figured out I was not copying the files and sourcing correctly.
- Thursday 4/7 - Started working on the concurrency exercise. I worked for about 4 hours reading on mutex variables and pthread conditionals and looking at examples of how to use them. I then looked at the class resources on the webpage and found the appropriate file needed for the random function.
- Monday 4/11 - I now have realized I should not have put this assignment off for so long and have been working for 6 hours finishing the concurrency exercise as well as this write up. I still have no clue how to do the git repo as a latex table so this number will probably increase. Edit: I worked for 3 more hours.

VI. Source Code

```
#include <stdio.h>
#include <pthread.h>
#include <time.h>
#include "mt19937ar.c"

#define MAX 32
pthread_mutex_t control;
pthread_cond_t conditionalCons, conditionalProd;

struct passData{
        int value;
        int wait;
};

struct passData buffer[MAX];
int bufferLength = 0;

/*-----------------------------------------
Function: consumer
Description: This function is what the
consumer thread will call. The function
checks to make sure the buffer isn't empty,
and if it is empty, will wait for the producer
to add a data set to the buffer.
-----------------------------------------*/

void* consumer(void *ptr)
{
        int i;
        while(1){
                pthread_mutex_lock(&control);
                while (bufferLength <= 0){
                        printf("Buffer␣is␣empty␣waiting␣for␣producer...␣\n");
                        pthread_cond_wait(&conditionalCons, &control);
                }
                pthread_mutex_unlock(&control);
                sleep(buffer[bufferLength-1].wait);

                pthread_mutex_lock(&control);
                printf("Consumer:␣Consumed␣item␣with␣value␣%d,␣Buffer␣at␣%d\n", buffer[
                bufferLength--;
                pthread_cond_signal(&conditionalProd);
                pthread_mutex_unlock(&control);
        }
        pthread_exit(0);
}

/*----------------------------------------
Function: producer
Description: This function is what the
producer thread will call. The function
checks to make sure the buffer isn't full,
and if it is empty, will wait for the
consumer to use a data set on the buffer
before adding a new one.
-----------------------------------------*/

void* producer(void *ptr)
{
        int i;
        int wait;
        while(1) {
                wait = (genrand_int32() % 5) + 3; //sleep for 3-7 seconds before produc
                sleep(wait);
                pthread_mutex_lock(&control);
                while(bufferLength == (MAX)){
                        printf("Buffer␣full␣waiting␣for␣consumer...␣\n");
                        pthread_cond_wait(&conditionalProd, &control);
                }
                buffer[bufferLength].wait = (genrand_int32() % 8) + 2;
                buffer[bufferLength].value = genrand_int32() % 100;
                bufferLength++;
                printf("Producer:␣Produced␣item␣with␣value␣%d.␣Buffer␣at␣%d␣\n", buffer
                pthread_cond_signal(&conditionalCons);
                pthread_mutex_unlock(&control);

        }
        pthread_exit(0);
}

int main(int argc, char **argv)
{
        pthread_t produce, consume;
        init_genrand(time(NULL));

        pthread_mutex_init(&control, NULL);
        pthread_cond_init(&conditionalCons, NULL);
        pthread_cond_init(&conditionalProd, NULL);


        pthread_create(&consume, NULL, consumer, NULL);
        pthread_create(&produce, NULL, producer, NULL);

        pthread_join(consume, NULL);
        pthread_join(produce, NULL);

        pthread_mutex_destroy(&control);
        pthread_cond_destroy(&conditionalCons);
        pthread_cond_destroy(&conditionalProd);

}
```

REFERENCES

[1] A. Liguori. (January) Qemu emulator user documentation. [Online]. Available: http://wiki.qemu.org/download/qemu-doc.html

[2] SpiderRico. (March) Consumer/producer with pthreads having waiting times. [Online]. Available: http://stackoverflow.com/questions/29317969/consumer-producer-with-pthreads-having-waiting-times