Homework#1 Group 35 Write-up

Yunfan Li, Zhicheng Fu

April 13, 2015

1 Command Logs

cd /scratch/spring2015/
mkdir cs444-group35
/scratch/bin/`acl_open` cs444-group35 wushu
/scratch/bin/`acl_open` cs444-group35 fuz
/scratch/bin/`acl_open` cs444-group35 dmcgrath
/scratch/bin/`acl_open` cs444-group35 lesliew
/scratch/bin/`acl_open` cs444-group35 moorsean

git clone git://git.yoctoproject.org/linux-yocto-3.14
git checkout tags/v3.14.24

git init

source /scratch/opt/environment-setup-i586-pocky-linux
cp /scratch/spring2015/files/ ./
cd files
qemu-system-i386 -gdb tcp::5535 -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”.
gdb
target remote :5535
continue
shutdown -h now

cd ../linux-yocto-3.14-3.14.24
cp /scratch/spring2015/files/config-3.14.26-yocto-qemu .config
make -j4 all
cd ../files
cp ../linux-yocto-3.14-3.14.24/arch/x86/bzImage .
qemu-system-i386 -nographic -kernel bzImage -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”.
shutdown -h now

cd ..
git add files
git add linux-yocto-3.14-3.14.24
git commit -m ”upload all files”
git push

2 Concurrency Solution

```
initialize buf_num, buf[32], pthread_mutex

create a thread with function A

while true

        get firstValue from random_number function
        get secondValue from random_number function

        while buf_num==32
                continue

        lock mutex

        buf[buf_num].firstValue = firstValue
        buf[buf_num].secondValue = secondValue
        buf_num <- 1 + buf_num

        unlock mutex

        get producer_sleep_time from random_number function

        sleep(producer_sleep_time)

function A (for pthread_create use)

        detach this thread

        while true
                while buf_num==0
                        continue

                lock mutex

                firstValue <- buf[0].firstValue
                secondValue <- buf[0].secondValue

                for i <- 0 to (buf_num-1)
                        buf[i].firstValue <- buf[i+1].firstValue
                        buf[i].secondValue <- buf[i+1].secondValue

                buf_num <- buf_num - 1

                unlock mutex

                sleep(secondValue)

                print firstValue
```

3 Git Log Table

47f9c34	Rasadell09	3 hours ago	Add Mersenne Twister files
6343b40	Rasadell09	28 hours ago	Add concurrency.c makefile for concurrency1
ddec9b8	SuzyWu2014	9 days ago	config file
10d793c	SuzyWu2014	9 days ago	add linux-yocto-3.14.24
a0bc9f5	SuzyWu2014	9 days ago	add bzImage envi_script
b57b624	SuzyWu2014	9 days ago	first commit

4 Work Log Table

	Work Log Sheet
Assignment:	CS-544 Homework #1
Members:	Zhicheng Fu , Yunfan Li

Start/Stop Time	Task Performed
14:00 - 18:00 4/1/2015	Build the kernel and run it in qemu on os-class
12:00 - 13:30 4/3/2015	Fix some bugs of the kernel
13:00 - 17:00 4/4/2015	Read books and learn materials
14:00 - 17:00 4/5/2015	review the knowledge of pthread in Linux
16:00 - 23:30 4/8/2015	Write the algorithm and write the structure of the pthread program
13:00 - 16:30 4/10/2015	Write the code of rdrand and Mersenne Twister
13:00 - 20:00 4/11/2015	Group write-up

Appendix 1

```
//Yunfan Li, Zhicheng Fu
//CS544-001
//Homework #1
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <pthread.h>
#include <libgen.h>
#include <math.h>
#include <time.h>
#include "mt19937ar.h"

struct item {
        int prt;
        int slp;
};
```

```
static pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;
int buf_num = 0;
struct item buf[32];

int asm_random(int begin, int end, int flag);

void *thread_func(void *arg);

int asm_random(int begin, int end, int flag)
{
        int result = 0;
        unsigned char ok;
        uint32_t rd;

        if (flag) {
                asm volatile ("rdrand %0; setc %1"
                              : "=r"(rd), "=qm"(ok)
                              );

                if (1 == ok) {
                        result = abs((int)rd);
                }
                else {
                        perror("Rdrand failed!\n");
                        exit(1);
                }
        }
        else {
                result = abs((int)genrand_int32());
        }

        if (begin != end) {
                result = begin + ( result%(end-begin+1) );
        }

        return result;
}

void *thread_func(void *arg)
{
        int c_num1 = 0;
        int c_num2 = 0;
        int i = 0;

        pthread_detach(pthread_self());

        while(1) {

                while(0 == buf_num) {
                        continue;
```

```
                }

                pthread_mutex_lock(&mtx);

                c_num1 = buf[0].prt;
                c_num2 = buf[0].slp;

                for(i = 0; i < (buf_num-1); i++) {
                        buf[i].prt = buf[i+1].prt;
                        buf[i].slp = buf[i+1].slp;
                }

                buf_num-=1;

                pthread_mutex_unlock(&mtx);

                sleep(c_num2);

                printf("%d\n", c_num1);
        }
}

int main(int argc, char **argv)
{
        int p_num1 = 0;
        int p_num2 = 0;
        int p_sleep_num = 0;

        pthread_t ptid;

        int flag = 0;
        unsigned int eax = 1;
        unsigned int ebx = 0;
        unsigned int ecx = 0;
        unsigned int edx = 0;

        asm volatile("cpuid"
                     : "=a"(eax), "=b"(ebx), "=c"(ecx), "=d"(edx)
                     : "a"(eax)
                     );

        if (ecx & 0x40000000) {
                flag = 1;
                printf("This machine supports RDRAND instruction.\n");
        }
        else {
                flag = 0;
                printf("This machine doesn't support RDRAND instruction.\n");
        }

        init_genrand(time(0));
```

```
        pthread_create(&ptid, NULL, thread_func, NULL);

        while(1) {
                p_num1 = asm_random(0, 0, flag);
                p_num2 = asm_random(2, 9, flag);

                while(32 == buf_num) {
                        continue;
                }

                pthread_mutex_lock(&mtx);

                buf[buf_num].prt = p_num1;
                buf[buf_num].slp = p_num2;

                buf_num+=1;

                pthread_mutex_unlock(&mtx);

                p_sleep_num = asm_random(3, 7, flag);

                sleep(p_sleep_num);
        }
}
```