George Suarez

CSE 460

Lab 6 – Thread Programming and Semaphore Part I XV6 Scheduling

1. Introduction to Thread Programming

*pthreads\_demo.cpp:*

// pthreads\_demo.cpp

#include <pthread.h>

#include <stdio.h>

using namespace std;

//The thread

void \*runner ( void \*data )

{

char \*tname = ( char \* )data;

printf("I am %s\n", tname );

pthread\_exit ( 0 );

}

int main ()

{

pthread\_t id1, id2; //thread identifiers

pthread\_attr\_t attr1, attr2; //set of thread attributes

char \*tnames[2] = { "Thread 1", "Thread 2" }; //names of threads

//get the default attributes

pthread\_attr\_init ( &attr1 );

pthread\_attr\_init ( &attr2 );

//create the threads

pthread\_create ( &id1, &attr1, runner, tnames[0] );

pthread\_create ( &id2, &attr2, runner, tnames[1] );

//wait for the threads to exit

pthread\_join ( id1, NULL );

pthread\_join ( id2, NULL );

return 0;

}

*Output of pthreads\_demo.cpp:*

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

**$** ./pthreads\_demo

I am Thread 1

I am Thread 2

*sdlthreads\_demo.cpp:*

#include <SDL/SDL.h>

#include <SDL/SDL\_thread.h>

#include <stdio.h>

using namespace std;

//The thread

int runner ( void \*data )

{

char \*tname = ( char \* )data;

printf("I am %s\n", tname );

return 0;

}

int main ()

{

SDL\_Thread \*id1, \*id2; //thread identifiers

char \*tnames[2] = { (char \*) "Thread 1", (char \*) "Thread 2" }; //names of threads

//create the threads

id1 = SDL\_CreateThread ( runner, tnames[0] );

id2 = SDL\_CreateThread ( runner, tnames[1] );

//wait for the threads to exit

SDL\_WaitThread ( id1, NULL );

SDL\_WaitThread ( id2, NULL );

return 0;

}

*Output of sdlthread\_demo.cpp:*

[006098556@csusb.edu@csevnc threads]$ ./sdlthread\_demo

I am Thread 1

I am Thread 2

*Modified pthreads\_demo.cpp:*

#include <pthread.h>

#include <stdio.h>

using namespace std;

//The thread

void \*runner ( void \*data )

{

char \*tname = ( char \* )data;

printf("I am %s\n", tname );

pthread\_exit ( 0 );

}

void \*runner2 ( void \*data )

{

char \*tname = (char \*)data;

printf("This is a different thread which is %s\n", tname);

pthread\_exit(0);

}

void \*runner3 ( void \*data )

{

char \*tname = (char \*)data;

printf("I am also a different thread which I am %s\n", tname);

pthread\_exit(0);

}

int main ()

{

pthread\_t id1, id2, id3; //thread identifiers

pthread\_attr\_t attr1, attr2, attr3; //set of thread attributes

char \*tnames[3] = { "Thread 1", "Thread 2", "Thread 3" }; //names of threads

//get the default attributes

pthread\_attr\_init ( &attr1 );

pthread\_attr\_init ( &attr2 );

pthread\_attr\_init ( &attr3 );

//create the threads

pthread\_create ( &id1, &attr1, runner, tnames[0] );

pthread\_create ( &id2, &attr2, runner2, tnames[1] );

pthread\_create ( &id3, &attr3, runner3, tnames[2] );

//wait for the threads to exit

pthread\_join ( id1, NULL );

pthread\_join ( id2, NULL );

pthread\_join ( id3, NULL );

return 0;

}

*Output of the modified pthreads\_demo.cpp:*

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

**$** ./pthreads\_demo

This is a different thread which is Thread 2

I am Thread 1

I am also a different thread which I am Thread 3

*Modified sdlthreads\_demo.cpp:*

#include <SDL2/SDL.h>

#include <SDL2/SDL\_thread.h>

#include <stdio.h>

using namespace std;

//The thread

int runner ( void \*data )

{

char \*tname = ( char \* )data;

printf("I am %s\n", tname );

return 0;

}

int runner2 ( void \*data )

{

char \*tname = ( char \* )data;

printf("Hello CSE 460! This is %s\n", tname );

return 0;

}

int runner3 ( void \*data )

{

char \*tname = ( char \* )data;

int result = 2 + 2;

printf("%s is executing 2 + 2 which is %d\n", tname, result);

return result;

}

int main ()

{

SDL\_Thread \*id1, \*id2, \*id3; //thread identifiers

//names of threads

char \*tnames[3] = { (char \*) "Thread 1", (char \*) "Thread 2", (char \*) "Thread 3" };

//create the threads

id1 = SDL\_CreateThread ( runner, "TestThread1", tnames[0] );

id2 = SDL\_CreateThread ( runner2, "TestThread2", tnames[1] );

id3 = SDL\_CreateThread ( runner3, "TestThread3" , tnames[2] ) ;

//wait for the threads to exit

SDL\_WaitThread ( id1, NULL );

SDL\_WaitThread ( id2, NULL );

SDL\_WaitThread ( id3, NULL );

return 0;

}

*Output of modified sdlthreads\_demo.cpp:*

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

**$** g++ -o sdlthreads\_demo sdlthreads\_demo.cpp -lSDL2 -lpthread

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

**$** ./sdlthreads\_demo

I am Thread 1

Hello CSE 460! This is Thread 2

Thread 3 is executing 2 + 2 which is 4

1. Unix Semaphore Facilities

Just went over the different functions that you can do with semaphores in the C programming language.

1. Using Semaphores

When executing the *sema1* program, this is what I see.

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

$ ./sema1 &

[1] 16268

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

**$** elelelelelelelelelel

16268 finished!

./ps auxw | grep sema1

georgesuarez 16296 0.0 0.0 4267752 872 s000 S+ 12:04PM 0:00.00 grep sema1

[1]+ Done ./sema1

The reason why it is outputting the chars ‘e’ and ‘l’ is because there are two variables that are assigned to the char values ‘e’ and ‘l’ which the program uses the functions *SEM\_DOWN()* and *SEM\_UP()* to see if the processes are safe to enter the critical section which then outputs the char values ‘e’ or ‘l’.

If the program is given any arguments, then it outputs

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

**$** ./sema1 a

ELELELELELELELELELEL

16360 finished!

The reason for this behavior is because the program checks if there are any additional arguments that are passed when running the program, and if there are then it will change the assignment of the char variables that were initialized in the beginning of the program to their capital letter versions.

*Sema1.cpp (modified):*

//sema1.cpp

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

#include <iostream>

#include <stdio.h>

using namespace std;

static int sem\_id; //semaphore id

//initializes semaphore using SETVAL

static int set\_semvalue(int val)

{

union semun sem\_union; // sem\_union;

sem\_union.val = val;

if (semctl(sem\_id, 0, SETVAL, sem\_union) == -1)

return (0);

return 1;

}

//delete semaphore

static int del\_semvalue()

{

union semun sem\_union; // sem\_union;

sem\_union.val = 1;

if (semctl(sem\_id, 0, IPC\_RMID, sem\_union) == -1)

return (0);

return 1;

}

static int SEM\_DOWN()

{

struct sembuf b;

b.sem\_num = 0;

b.sem\_op = -1; //P(), i.e. down()

b.sem\_flg = SEM\_UNDO;

if (semop(sem\_id, &b, 1) == -1)

{

cout << "Semaphore DOWN() failed!" << endl;

return 0;

}

return 1;

}

static int SEM\_UP()

{

struct sembuf b;

b.sem\_num = 0;

b.sem\_op = 1; //V(), i.e. UP()

b.sem\_flg = SEM\_UNDO;

if (semop(sem\_id, &b, 1) == -1)

{

cout << "Semaphore UP() failed!" << endl;

return 0;

}

return 1;

}

int main(int argc, char \*argv[])

{

int i, pause\_time;

char ce = 'e', cl = 'l';

srand((unsigned int)getpid()); //seed RNG with process id

sem\_id = semget((key\_t)1234, 1, 0666 | IPC\_CREAT);

if (argc > 0)

{

if (!set\_semvalue(1))

{ //process can enter CS

cout << "Semaphore initialized failed!" << endl;

exit(EXIT\_FAILURE);

}

if (argc > 1)

{

ce = 'E';

cl = 'L';

}

sleep(1);

}

else

{

if (!set\_semvalue(0))

{ //process will be blocked initially

cout << "Semaphore initialized failed!" << endl;

exit(EXIT\_FAILURE);

}

sleep(1);

}

//enter and leave critical section 10 times

if (strcmp(argv[1], "1") == 0)

{

for (i = 0; i < 10; i++)

{

if (!SEM\_DOWN())

exit(EXIT\_FAILURE);

cout << ce;

fflush(stdout); //entering critical section

pause\_time = rand() % 3; //simulate critical section

sleep(pause\_time);

cout << cl;

fflush(stdout); //leaving critical section

if (!SEM\_UP())

exit(EXIT\_FAILURE); //signal other waiting process

pause\_time = rand() % 2;

sleep(pause\_time);

}

cout << endl

<< getpid() << " finished!" << endl;

if (argc > 0)

{

sleep(2);

del\_semvalue();

}

}

else if (strcmp(argv[1], "0") == 0)

{

cout << "I am going to wait forevr..." << endl;

while (1)

{

}

}

exit(EXIT\_SUCCESS);

}

*Outputs of sema1.cpp (mod):*

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

**$** ./sema1 1

ELELELELELELELELELEL

28528 finished!

**georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 6 on master\***

**$** ./sema1 0

I am going to wait forever...

1. XV6 Scheduling

[006098556@jb359-16 xv6]$ make qemu-nox

which: no qemu in (/usr/local/bin:/opt/eclipse:/opt/scilab/bin:/opt/android-studio/bin:/opt/argouml:/usr/lib64/openmpi/bin:/usr/local/cuda/bin:/share/bin:/opt/Xilinx/14.7/ISE\_DS/ISE/bin/lin64:/opt/Xilinx/14.7/ISE\_DS/common/bin/lin64:/opt/android-sdk-linux/tools:/opt/android-sdk-linux/platform-tools:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/u/cse/006098556/bin/)

qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512

(process:10443): GLib-WARNING \*\*: gmem.c:482: custom memory allocation vtable not supported

xv6...

cpu1: starting

cpu0: starting

Process initcode with pid 1 running

Process initcode with pid 1 running

sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

Process init with pid 1 running

inProcess init with pid 1 running

it: starting sh

Process init with pid 1 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

$ foo 4

Process sh with pid 2 running

Process sh with pid 2 running

Process sh with pid 3 running

exec foo failed

Process sh with pid 2 running

$ foo 4

Process sh with pid 2 running

Process sh with pid 4 running

eProcess sh with pid 4 running

xec foo failed

Process sh with pid 2 running

$ q

Process sh with pid 2 running

Process sh with pid 5 running

Process sh with pid 5 running

exec q failed

Process sh with pid 2 running

$ quit

Process sh with pid 2 running

Process sh with pid 6 running

Process sh with pid 6 running

exec quit failed

Process sh with pid 2 running

$

Process sh with pid 2 running

Process sh with pid 7 running

Process sh with pid 7 running

exec failed

Process sh with pid 2 running

Changing *proc.c:*

[006098556@jb359-15 xv6]$ make qemu-nox

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o proc.o proc.c

ld -m elf\_i386 -T kernel.ld -o kernel entry.o bio.o console.o

exec.o file.o fs.o ide.o ioapic.o kalloc.o kbd.o lapic.o log.o

main.o mp.o picirq.o pipe.o proc.o sleeplock.o spinlock.o

string.o swtch.o syscall.o sysfile.o sysproc.o timer.o trapasm.o

trap.o uart.o vectors.o vm.o -b binary initcode entryother

objdump -S kernel > kernel.asm

objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .\* / /; /^$/d' >

kernel.sym

dd if=/dev/zero of=xv6.img count=10000

10000+0 records in

10000+0 records out

5120000 bytes (5.1 MB) copied, 0.116467 s, 44.0 MB/s

dd if=bootblock of=xv6.img conv=notrunc

1+0 records in

1+0 records out

512 bytes (512 B) copied, 0.00122226 s, 419 kB/s

dd if=kernel of=xv6.img seek=1 conv=notrunc

357+1 records in

357+1 records out

183072 bytes (183 kB) copied, 0.00416878 s, 43.9 MB/s

which: no qemu in

(/usr/local/bin:/opt/eclipse:/opt/scilab/bin:/opt/androidstudio/bin:/opt/argouml:/usr/lib64/openmpi/bin:/usr/local/cuda/b

in:/share/bin:/opt/Xilinx/14.7/ISE\_DS/ISE/bin/lin64:/opt/Xilinx/

14.7/ISE\_DS/common/bin/lin64:/opt/android-sdklinux/tools:/opt/android-sdk-linux/platformtools:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/u/cse/0

04867222/bin/.)

qemu-system-i386 -nographic -drive

file=fs.img,index=1,media=disk,format=raw -drive

file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512

(process:26800): GLib-WARNING \*\*: gmem.c:482: custom memory

allocation vtable not supported

xv6...

cpu1: starting

cpu0: starting

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2

inodestart 32 bmap start 58

Process initcode with pid 1 running [06098556@jb359-15 xv6]$ make qemu-nox

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o proc.o proc.c

ld -m elf\_i386 -T kernel.ld -o kernel entry.o bio.o console.o

exec.o file.o fs.o ide.o ioapic.o kalloc.o kbd.o lapic.o log.o

main.o mp.o picirq.o pipe.o proc.o sleeplock.o spinlock.o

string.o swtch.o syscall.o sysfile.o sysproc.o timer.o trapasm.o

trap.o uart.o vectors.o vm.o -b binary initcode entryother

objdump -S kernel > kernel.asm

objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .\* / /; /^$/d' >

kernel.sym

dd if=/dev/zero of=xv6.img count=10000

10000+0 records in

10000+0 records out

5120000 bytes (5.1 MB) copied, 0.116467 s, 44.0 MB/s

dd if=bootblock of=xv6.img conv=notrunc

1+0 records in

1+0 records out

512 bytes (512 B) copied, 0.00122226 s, 419 kB/s

dd if=kernel of=xv6.img seek=1 conv=notrunc

357+1 records in

357+1 records out

183072 bytes (183 kB) copied, 0.00416878 s, 43.9 MB/s

which: no qemu in

(/usr/local/bin:/opt/eclipse:/opt/scilab/bin:/opt/androidstudio/bin:/opt/argouml:/usr/lib64/openmpi/bin:/usr/local/cuda/b

in:/share/bin:/opt/Xilinx/14.7/ISE\_DS/ISE/bin/lin64:/opt/Xilinx/

14.7/ISE\_DS/common/bin/lin64:/opt/android-sdklinux/tools:/opt/android-sdk-linux/platformtools:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/u/cse/0

04867222/bin/.)

qemu-system-i386 -nographic -drive

file=fs.img,index=1,media=disk,format=raw -drive

file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512

(process:26800): GLib-WARNING \*\*: gmem.c:482: custom memory

allocation vtable not supported

xv6...

cpu1: starting

cpu0: starting

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2

inodestart 32 bmap start 58

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process initcode with pid 1 running

Process init with pid 1 running

init: starting sh

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

Process init with pid 2 running

$ foo

Process sh with pid 2 running

Process sh with pid 3 running

Process sh with pid 3 running

Process sh with pid 3 running

Process sh with pid 3 running

Process sh with pid 3 running

Process sh with pid 3 running

Process sh with pid 3 running

Process sh with pid 3 running

Process foo with pid 3 running

Parent 3 creating child 4

Process foo with pid 4 running

Process foo with pid 4 running

Child 4 created

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 4 running

Process foo with pid 3 running

Process foo with pid 3 running

Process foo with pid 3 running

Process foo with pid 3 running

Process foo with pid 3 running

Process foo with pid 3 running

Process foo with pid 3 running

Process foo with pid 3 running

Process init with pid 1 running

zombie!

Process sh with pid 2 running

Adding The Time Stamp

[006098556@jb359-2 xv6]$ make qemu-nox

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o console.o console.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o exec.o exec.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o fs.o fs.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o ide.o ide.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o lapic.o lapic.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o main.o main.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o mp.o mp.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o pipe.o pipe.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o proc.o proc.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o sleeplock.o sleeplock.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o spinlock.o spinlock.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o syscall.o syscall.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o sysfile.o sysfile.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o sysproc.o sysproc.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o trap.o trap.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o uart.o uart.c

gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall

-MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stackprotector

-c -o vm.o vm.c

ld -m elf\_i386 -T kernel.ld -o kernel entry.o bio.o console.o

exec.o file.o fs.o ide.o ioapic.o kalloc.o kbd.o lapic.o log.o

main.o mp.o picirq.o pipe.o proc.o sleeplock.o spinlock.o

string.o swtch.o syscall.o sysfile.o sysproc.o timer.o trapasm.o

trap.o uart.o vectors.o vm.o -b binary initcode entryother

objdump -S kernel > kernel.asm

objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .\* / /; /^$/d' >

kernel.sym

dd if=/dev/zero of=xv6.img count=10000

10000+0 records in

10000+0 records out

5120000 bytes (5.1 MB) copied, 0.116044 s, 44.1 MB/s

dd if=bootblock of=xv6.img conv=notrunc

1+0 records in

1+0 records out

512 bytes (512 B) copied, 0.000914179 s, 560 kB/s

dd if=kernel of=xv6.img seek=1 conv=notrunc

357+1 records in

357+1 records out

183072 bytes (183 kB) copied, 0.00410289 s, 44.6 MB/s

which: no qemu in

(/usr/local/bin:/opt/eclipse:/opt/scilab/bin:/opt/androidstudio/bin:/opt/argouml:/usr/lib64/openmpi/bin:/usr/local/cuda/b

in:/share/bin:/opt/Xilinx/14.7/ISE\_DS/ISE/bin/lin64:/opt/Xilinx/

14.7/ISE\_DS/common/bin/lin64:/opt/android-sdklinux/tools:/opt/android-sdk-linux/platformtools:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/u/cse/0

04867222/bin/.)

qemu-system-i386 -nographic -drive

file=fs.img,index=1,media=disk,format=raw -drive

file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512

(process:28641): GLib-WARNING \*\*: gmem.c:482: custom memory

allocation vtable not supported

xv6...

cpu1: starting

cpu0: starting

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

Process initcode with pid 1 running with createTime 0

**Discussion:** The lab overall wasn’t too hard until I had to figure out what to modify in the sema1 program, but I still manage to get it working. The XV6 took a while since I couldn’t login through SSH, so I had to use the lab computer to finish this part. In the end, I manage to finish all the parts in this lab successfully. I will give myself **20/20 points.**