Lab – III

"XV6 Threads"

Kiran Kumar Gangadevi

Initially, to start off with the project,

We have created a clone() system call: **int clone(void *stack, int size**).

The clone() creates a new kernel thread. It is a modification of fork() system call and uses the parent's address space and uses the same file descriptor. The function returns in the stack of the current thread.

It returns the pid of the child to the parent and 0 to the newly created thread.

For implementing the system call. We made the following changes in the files.

We have written the main implementation in **proc.c**

In **syscall.h** the position of the system call vector that connects it to system call implementation is defined and we connect the code to kernel interface.

In **user.h** we added the prototype for clone

We defined the function prototype in **syscall.c**. We added the system call to the intialiaztion list of syscall array.

We also modified **defs.h** and **usys.S** to include clone()

We added the new program to UPROG maccros list in **Makefile** and also thread_lib in ULIB.

We created a thread library using **thread_lib.c** which creates threads using thread_create() function that uses clone() to create child. We calculated the stacksize and created pointers that point to parent process' stack using C calling convention.

We implemented a simple lock. Using two functions lock_acquire() and lock_release(), allows to acquired and release the lock. Lock_init() function intializes the lock.

For testing the simple lock:

We created a test program **threadtest.c** that accepts the user's input through command line argument. The user inputs the number of threads and the number of passes that each thead must execute. The test program creates the specified number of threads. For each thread, it makes a copy of address space acquires the lock and then prints out a statement to show that it is excecuting. We maintain an counter that counts the total number of processes that is being created.

We also implemented Anderson's Lock. Using two functions acquire_andersonlock() and release_andersonlock(), allows to acquired and release the lock. init_andersonlock() function intializes the lock.

For testing the Anderson Lock:

We created a test program **threadtest_al.c** that accepts the user's input through command line argument. The user inputs the number of threads and the number of passes that each thead must execute. The test program creates the specified number of threads. For each thread, it makes a copy of address space acquires the lock and then prints out a statement to show that it is excecuting.

Inside thread_lib.c there are implementations for thread_create, lock_acquire and lock_release functions.

We created a thread library header file.

List of files Modified:

proc.c

syscall.c

syscall.h

Makefile

defs.h

usys.S

user.h

thread_lib.c

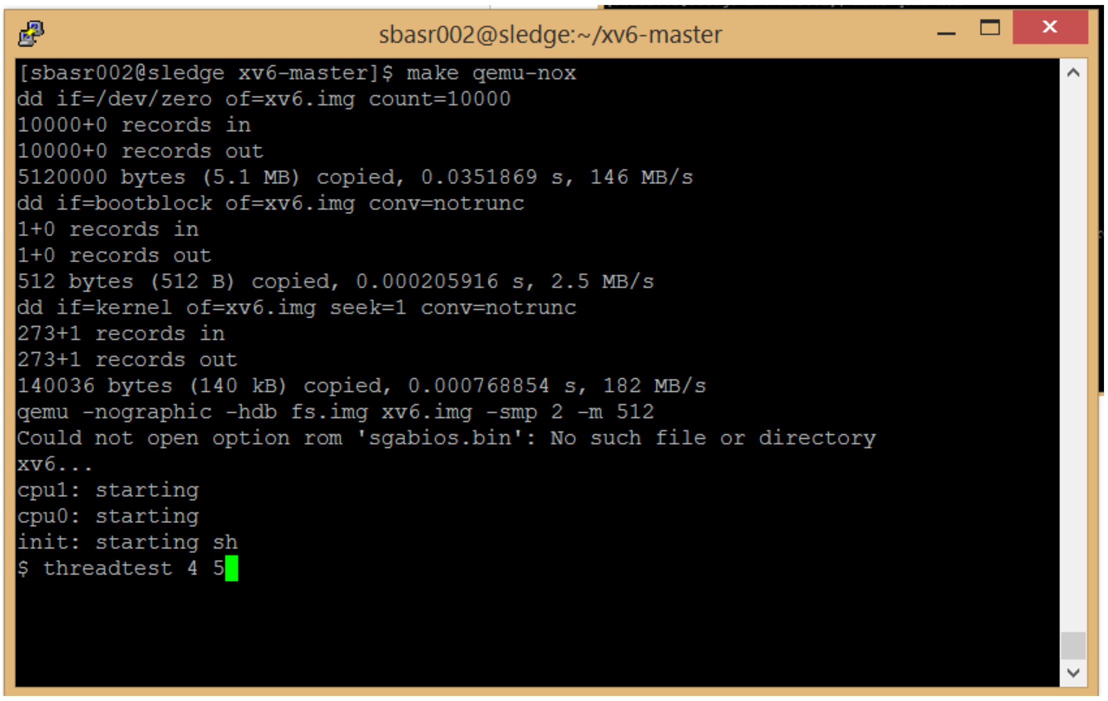
thread_lib.h

Test program:

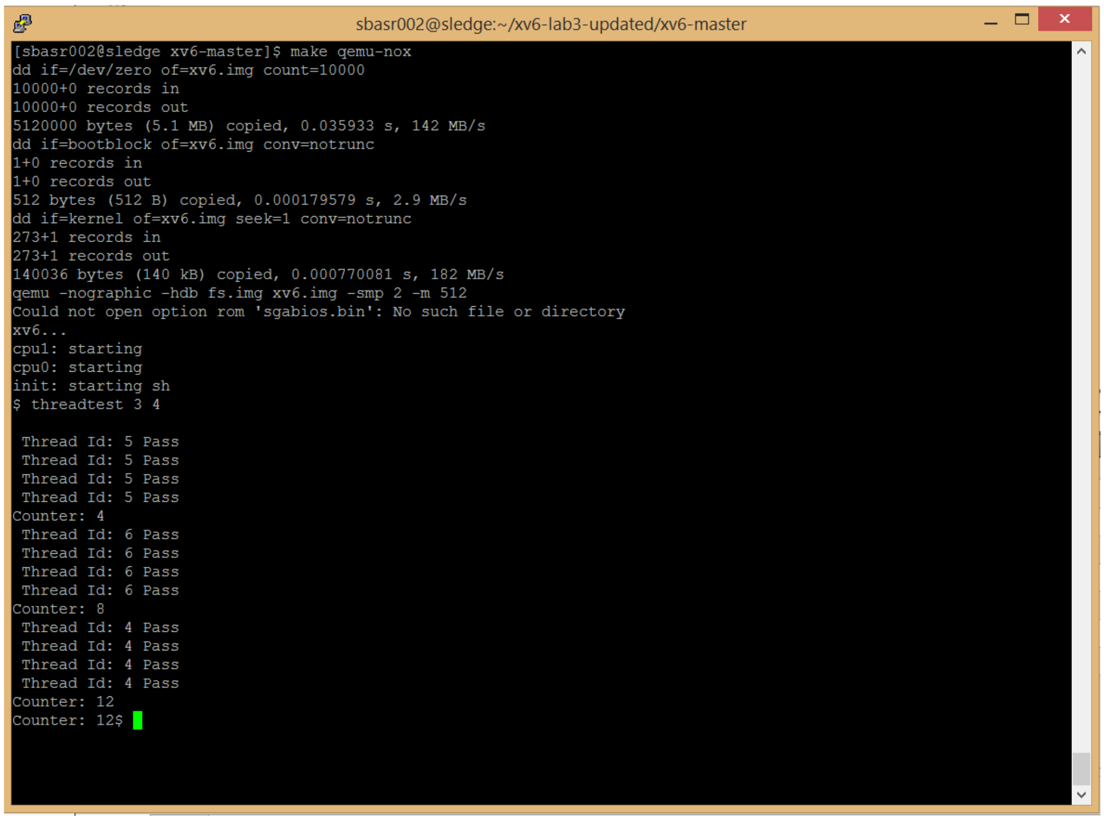
threadtest.c – For testing the simple lock

threadtest_al.c – For testing Anderson's Lock

Simple Lock Testing:



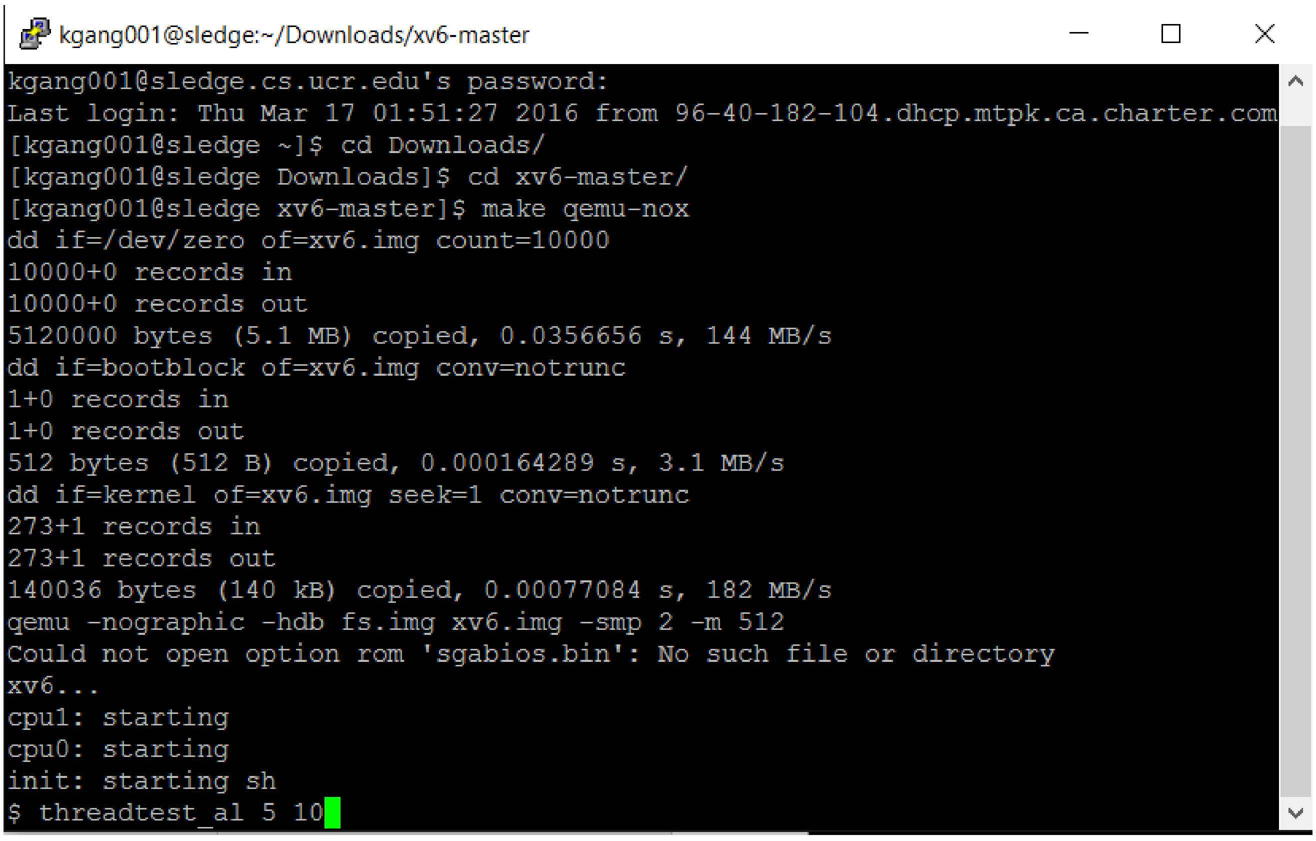
```
[sbasr002@sledge xv6-master]$ make qemu-nox
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB) copied, 0.0351869 s, 146 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000205916 s, 2.5 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
273+1 records in
273+1 records out
140036 bytes (140 kB) copied, 0.000768854 s, 182 MB/s
qemu -nographic -hdb fs.img xv6.img -smp 2 -m 512
Could not open option rom 'sgabios.bin': No such file or directory
xv6...
cpu1: starting
cpu0: starting
init: starting sh
$ threadtest 4 5
```



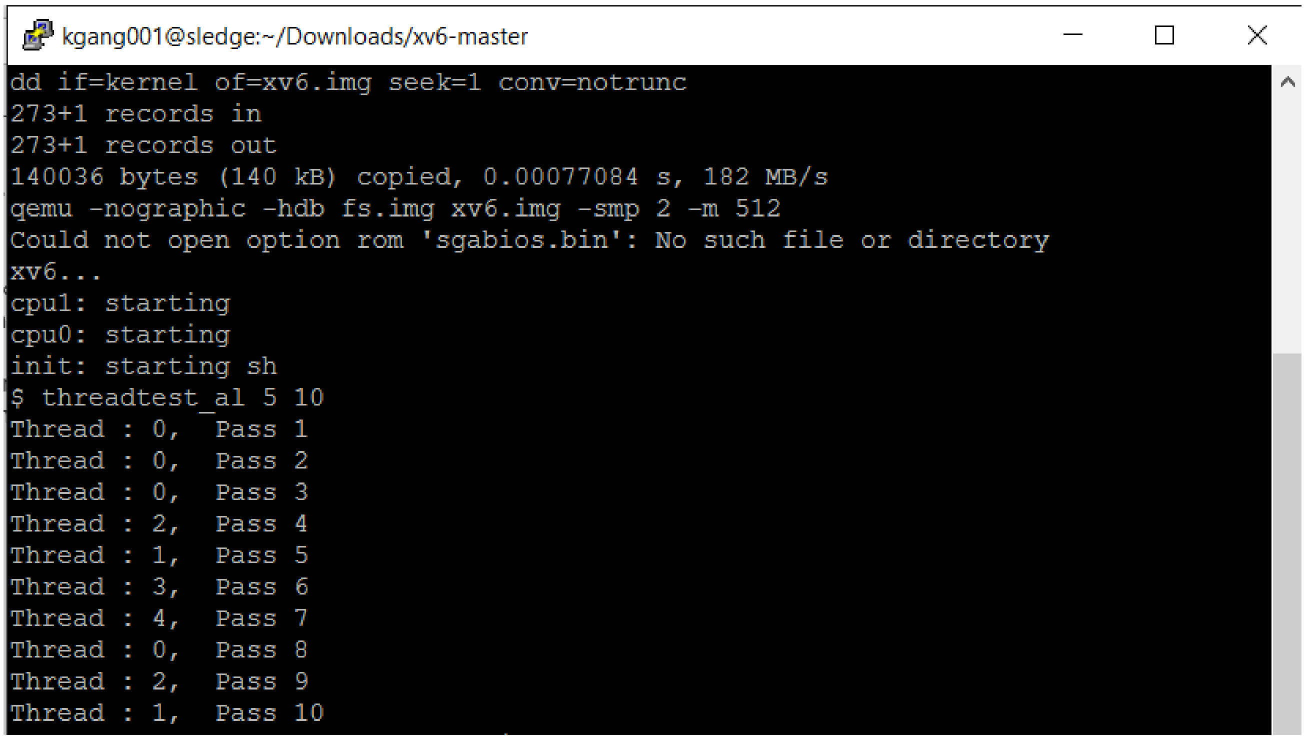
```
[sbasr002@sledge xv6-master]$ make qemu-nox
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB) copied, 0.035933 s, 142 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000179579 s, 2.9 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
273+1 records in
273+1 records out
140036 bytes (140 kB) copied, 0.000770081 s, 182 MB/s
qemu -nographic -hdb fs.img xv6.img -smp 2 -m 512
Could not open option rom 'sgabios.bin': No such file or directory
xv6...
cpu1: starting
cpu0: starting
init: starting sh
$ threadtest 3 4

 Thread Id: 5 Pass
 Thread Id: 5 Pass
 Thread Id: 5 Pass
 Thread Id: 5 Pass
Counter: 4
 Thread Id: 6 Pass
 Thread Id: 6 Pass
 Thread Id: 6 Pass
 Thread Id: 6 Pass
Counter: 8
 Thread Id: 4 Pass
 Thread Id: 4 Pass
 Thread Id: 4 Pass
 Thread Id: 4 Pass
Counter: 12
Counter: 12$
```

Anderson Lock Testing:


kgang001@sledge:~/Downloads/xv6-master

```
kgang001@sledge.cs.ucr.edu's password:
Last login: Thu Mar 17 01:51:27 2016 from 96-40-182-104.dhcp.mtpk.ca.charter.com
[kgang001@sledge ~]$ cd Downloads/
[kgang001@sledge Downloads]$ cd xv6-master/
[kgang001@sledge xv6-master]$ make qemu-nox
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB) copied, 0.0356656 s, 144 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000164289 s, 3.1 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
273+1 records in
273+1 records out
140036 bytes (140 kB) copied, 0.00077084 s, 182 MB/s
qemu -nographic -hdb fs.img xv6.img -smp 2 -m 512
Could not open option rom 'sgabios.bin': No such file or directory
xv6...
cpu1: starting
cpu0: starting
init: starting sh
$ threadtest_al 5 10
```


kgang001@sledge:~/Downloads/xv6-master

```
dd if=kernel of=xv6.img seek=1 conv=notrunc
273+1 records in
273+1 records out
140036 bytes (140 kB) copied, 0.00077084 s, 182 MB/s
qemu -nographic -hdb fs.img xv6.img -smp 2 -m 512
Could not open option rom 'sgabios.bin': No such file or directory
xv6...
cpu1: starting
cpu0: starting
init: starting sh
$ threadtest_al 5 10
Thread : 0,  Pass 1
Thread : 0,  Pass 2
Thread : 0,  Pass 3
Thread : 2,  Pass 4
Thread : 1,  Pass 5
Thread : 3,  Pass 6
Thread : 4,  Pass 7
Thread : 0,  Pass 8
Thread : 2,  Pass 9
Thread : 1,  Pass 10
```