Report: Project 2

Denny Shen

dennyshe@usc.edu

ID: 2491547502

- Describe changes to the xv6. Name the files your modified, briefly describe the changes.

syscall.c, sysproc.c (sycall.h, user.h… were already updated):

Updated according functions and linkers for the added syscalls of thread and mutex.

proc.h:

I added structure codes for Mutex.

I added a TBLOCKED state for thread states.

I added a lock in the proc struct to synchronize threads within this process.

kthread.h:

I added syscall functions for kernel thread.

proc.c:

I added 4 new functions for thread: create, id, exit, join.

I updated 2 existing process functions, growproc, exit, so that they now work with threads. The changes are mostly adding locks and helper function, kill_all().

I added a new struct for a new spinlock to lock the mutex table.

exec.c:

I updated 1 existing process function, exec, with a helper function, kill_others(), so that it now works with threads.

Clearly explain the changes you made as a part of Task1.1 to support:

1. synchronization to shared fields

I have implemented the thread table differently by adding a spinlock in the proc struct.

```
// Per-process state
struct proc {
  uint sz;                     // Size of process memory (bytes)
  pde_t* pgdir;                // Page table
  enum procstate state;        // Process state
  int pid;                     // Process ID
  struct proc *parent;         // Parent process
  int killed;                  // If non-zero, have been killed
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
  struct thread threads[NTHREAD]; // static thread array
  struct spinlock lock;        // Synchronize threads within this process
};
```

By acquiring this lock using acquire(&proc->lock);, the threads within the process could be synchronized since there can only be one thread holding onto this process specific lock at a time.

Then the normal usage of ptable, mtable to allow only one process entering the critical section is implemented with the tradition style.

2. expected behavior of existing system calls

For growproc(), I acquired ptable lock to protect sz variable, and release the lock before return.

There should be no change added to fork() as there is no conflicts on shared variables, and the existing lock is good enough.

For exit(), since all threads should be killed before exiting a process, I added a kill_all() helper function to kill all threads before exiting the process.

For exec(), similar to exit(), I added a kill_others() to kill all other threads and keep the current thread for execution.

Screenshot on Success:

threadtest1:

```
student@studentVM:~/CSCI350_Projects/project-2-dennyshe/xv6-kernel-threads$ 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, 4.9 MiB) copied, 0.0519237 s, 98.6 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.00927821 s, 55.2 kB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
366+1 records in
366+1 records out
187868 bytes (188 kB, 183 KiB) copied, 0.00348429 s, 53.9 MB/s
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 1 -m 512
xv6...
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2          inodestart 32 bmap start 58
init: starting sh
$ threadtest1
thread in main 3,process 3
Thread id is: 4
Thread id is: 5
Thread is exiting.
Got id : 4
Thread is exiting.
Got id : 5
Finished.
$
```

threadtest2:

```
student@studentVM:~/CSCI350_Projects/project-2-dennyshe/xv6-kernel-threads$ 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, 4.9 MiB) copied, 0.057529 s, 89.0 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.000128089 s, 4.0 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
366+1 records in
366+1 records out
187868 bytes (188 kB, 183 KiB) copied, 0.00396085 s, 47.4 MB/s
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 1 -m 512
xv6...
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2          inodestart 32 bmap start 58
init: starting sh
$ threadtest2
~~~~~~~~~~~~~~~~~~ thread test ~~~~~~~~~~~~~~~~~~
thread 1 says hello
thread 2 says hello
thread 3 says hello
all threads exited
$
```

threadtest3:

```
student@studentVM:~/CSCI350_Projects/project-2-dennyshe/xv6-kernel-threads$ 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, 4.9 MiB) copied, 0.0473086 s, 108 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.000131016 s, 3.9 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
366+1 records in
366+1 records out
187868 bytes (188 kB, 183 KiB) copied, 0.00630369 s, 29.8 MB/s
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 1 -m 512
xv6...
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2          inodestart 32 bmap start 58
init: starting sh
$ threadtest3
Joining 4
Thread 4 running !
Thread 5 running !
Thread 6 running !
Joining 5
Joining 6

All threads done!
$
```

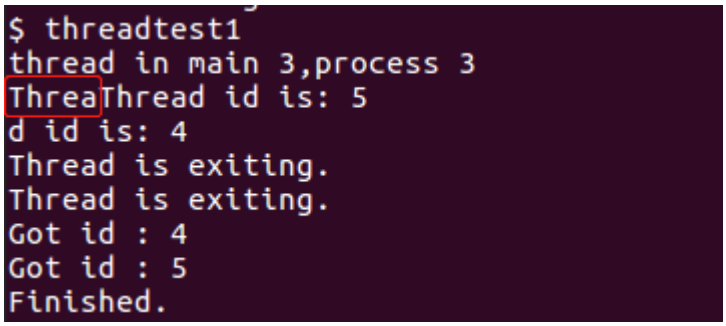
mutextest1:

```
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 1 -m 512
xv6...
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2          inodestart 32 bmap start 58
init: starting sh
$ mutextest1
~~~~~~~~~~~~~~~~~~ mutex test 1 ~~~~~~~~~~~~~~~~~~
joining on thread 4
thread 4 said hi
finished join
joining on thread 5
thread 5 said hi
finished join
joining on thread 6
thread 6 said hi
finished join
joining on thread 7
thread 7 said hi
finished join
joining on thread 8
thread 8 said hi
finished join
joining on thread 9
thread 9 said hi
finished join
joining on thread 10
thread 10 said hi
finished join
joining on thread 11
thread 11 said hi
finished join
joining on thread 12
thread 12 said hi
finished join
joining on thread 13
thread 13 said hi
finished join
joining on thread 14
thread 14 said hi
finished join
joining on thread 15
thread 15 said hi
finished join
joining on thread 16
thread 16 said hi
finished join
joining on thread 17
thread 17 said hi
finished join
joining on thread 18
thread 18 said hi
finished join
Finished.
$
```

mutextest2:

```
student@studentVM:~/CSCI350_Projects/project-2-dennyshe/xv6-kernel-threads$ 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, 4.9 MiB) copied, 0.0463783 s, 110 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.000133208 s, 3.8 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
367+1 records in
367+1 records out
187940 bytes (188 kB, 184 KiB) copied, 0.00353201 s, 53.2 MB/s
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 1 -m 512
xv6...
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2          inodestart 32 bmap start 58
init: starting sh
$ mutextest2
~~~~~~~~~~~~~~~~~~ mutex test 2 ~~~~~~~~~~~~~~~~~~
test passed
$
```

Weird Printing:



The print statement could appear in weird sequence. The thread was probably switched out during the print statement at the place indicated by the red box. This should be normal. (This might happen frequently in threadtest3 as I actually tried multiple times to acquire the exact same output as the expected one)