CS 350 - Operating Systems **Winter 2018**

Lecture 5: January 17, 2018

Lecturer: Lesley Istead *Notes By: Harsh Mistry*

5.1 Synchronization Continued

5.1.1 Spin Locks in OS/161

```
struct spinlock {
  volatile spinlock_data_t lk_lock;
  struct cpu *lk_holder;
};

void spinlock_init(struct spinlock *lk}
void spinlock_acquire(struct spinlock *lk);
void spinlock_release(struct spinlock *lk);
```

`spinlock_acquire` calls `spinlock_data_testandset` in a loop until the lock is acquired.

Example 5.1 *Spin-lock implementation for MIPS using in line assembly*

```
/* return value 0 indicates lock was acquired */
spinlock_data_testandset(volatile spinlock_data_t *sd)
{
  spinlock_data_t x,y;
  y = 1;
  __asm volatile(
    ".set push;"      /* save assembler mode */
    ".set mips32;"    /* allow MIPS32 instructions */
    ".set volatile;"  /* avoid unwanted optimization */
    "ll %0, 0(%2);"   /*   x = *sd */
    "sc %1, 0(%2);"   /*   *sd = y; y = success? */
    ".set pop"        /* restore assembler mode */
    : "=r" (x), "+r" (y) : "r" (sd));
  if (y == 0) { return 1; }
  return x;
}
```

5.1.2 Thread Blocking

- Sometimes a thread will wait for something e.g
  - wait for lock to be released

  - wait for data from a (relatively) slow device
  - wait for input from a keyboard
  - wait for busy device to become idle

- When a thread blocks, its stops running:
  - the scheduler chooses a new thread to run
  - a context switch from teh blocking thread to the new thread occurs
  - the blocking thread is queued in a `wait queue` (not on the ready list)
  - Eventually, a blocked thread is signalled and awakened by another thread

5.1.3 Wait Channels in OS/161

- Wait channels are used to implement thread blocking in OS/161
  - `void wchan_sleep(struct whan *wc);`, blocks calling a thread on wait channel and causes a context switch
  - `void wchan_wakeall(struct wchan *wc);`, unblock all threads sleeping on wait channel `wc`
  - `void wchan_wakeone(struct wchan *wc);`, unblock one thread sleeping on wait channel `wc`
  - `void wchan_lock(struct wchan *wc);`, prevents operations on wait channel `wc`
- There can also be many different wait channels, holding thread that are blocked for different reasons.

5.1.4 OS/161 Lock Implementation

Pseudo Code For A1 Q1

Acquire :

```
spin_acquire (lock->spin)
KASSERT(!lockowner)
KASSERT(!lock == null)
while (lock->held) {
  wchan_lock(lock->wc);
  spin_release(lock->wc);
  wchan_sleep(lock->wc);
  spin_aquire(lock->spin)
}
lock->held = 1;
lock->owner - curthread;
spin_release(lock->spin);
```

Release :

```
KASSERT(Owns the block)
spin_acquire(lock->spin);
lock->held = 0;
lock->owner = null;
wchan_wakeone(lock->wc);
spin_release(lock->spin)
```