# COMP3500: Project 3 - Part 4 Condition Variables

**Exercise 1:** How to modify the following Lock struct in OS/161? Can you complete the following lock function prototypes?

/\* kern/include/synch.h \*/

struct lock {

char \*name;

/\* add what you need here. How? \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

};

struct lock \*lock\_create (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­­­­­);

void lock\_acquire (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

void lock\_release (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

int lock\_do\_i\_hold(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

void lock\_destroy (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

**Exercise 2:** Please complete the following lock\_acquire()

void lock\_acquire(struct lock \*lock) {

turn off \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) /\* check deadlock \*/

panic("lock %s at %p: Deadlock.\n", lock->name, lock);

/\* wait the lock to become free \*/

while (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {

sleep this thread; /\*see P(\*sem) in next slide\*/

}

/\* this thread is holding the lock \*/

lock’s holder is set to \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

turn on interrupts to \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ level;

}

**Exercise 3:** Please complete the lock\_do\_i\_hold() function.

int lock\_do\_i\_hold(struct lock \*lock) {

int spl, same;

use assert() to input argument lock;

Turn off interrupts;

if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

set same to 1; /\* true \*/

else set same to 0; /\* false \*/

Turn on interrupts to previous level;

/\* 1 means lock is held by this thread \*/

return same;

}

**Exercise 4:** Can you explain this code?Which lock\_acqure() and lock\_release() are a pair? Can we improve this code using wait and signal?

char consumer() {

char c;

lock\_acquire(mutex);

while (count == 0) {

lock\_release(mutex);

lock\_acquire(mutex);

}

count--;

c = buffer[tail];

tail++;

if (tail == SIZE) {

tail = 0;

}

lock\_release(mutex);

return c;

}

void producer(char c) {

lock\_acquire(mutex);

while (count == SIZE) {

lock\_release(mutex);

lock\_acquire(mutex);

}

count++;

buffer[head] = c;

head++;

if (head == SIZE) {

head = 0;

}

lock\_release(mutex);

}

**Exercise 5:** Please complete the consumer code using condition variables.

char consumer() {

char c;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

while (count == 0) {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

count--;

c = buffer[tail];

tail++;

if (tail == SIZE) {

tail = 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return c;

}

void producer(char c) {

lock\_acquire(mutex);

while (count == SIZE) {

cv\_wait(notFull, mutex);

}

count++;

buffer[head] = c;

head++;

if (head == SIZE) {

head = 0;

}

cv\_signal(notEmpty, mutex);

lock\_release(mutex);

}