# Operating Systems
# ECE344

## *Lab assignment 1: Synchronization*

Ding Yuan

# Overview of the lab assignment

- Task 1: implementing synchronization primitives
  - 1a: implement lock
  - 1b: implement condition variable
- Task 2: use synchronization primitives to solve problems
  - 2a: Mice and cats
  - 2b: traffic lights

# Lock and cond. var.

```
P(sem) {
  Disable interrupts;
  while (sem->count == 0) {
    thread_sleep(sem); /* current thread
                 will sleep on this sem */
  }
  sem->count--;
  Enable interrupts;
}
```

```
V(sem) {
  Disable interrupts;
  sem->count++;
  thread_wakeup (sem); /* this will wake
      up all the threads waiting on this
      sem. Why wake up all threads? */
  Enable interrupts;
}
```

- Needs atomic region
  - Atomic region can be done in a similar way to semaphore
- If you understand how semaphore is implemented, should be trivial!
  - Cannot use semaphore to implement lock or cond. var.

# Synchronization problems

- How to start?
  - First: write operation code
  - Next: carefully reason about all the possible interleaving and timing scenarios
  - Add synchronization

# Mice and cats

- Two bowls, multiple cats and mice
- Safety criteria:
  - If a cat is eating at either dishes, no mouse can eat
  - If a mouse is eating at either dishes, no cat can eat

# Operation code

```
void sem_eat(const char *who, int num, int bowl, int iteration) {
  kprintf("%s: %d starts eating: bowl %d, iteration %d\n", who, num,
           bowl, iteration);
  clocksleep(1);
  kprintf("%s: %d ends eating: bowl %d, iteration %d\n", who, num,
              bowl, iteration);
}

void mousesem(void * p, unsigned long mousenumber) {
  int bowl, iteration;
  for (iteration = 0; iteration < 4; iteration++) {
    sem_eat (”mouse", mousenumber, bowl, iteration);
  }
}

int catmousesem(.. ..) {
 for (index = 0; index < NCATS; index++)
   thread_fork("catsem Thread”, NULL, index, catsem, NULL);

 for (index = 0; index < NMICE; index++)
   thread_fork(”mousesem Thread”, NULL, index, mousesem, NULL);
}
```

# About starvation

- You do not need to consider priority or starvation
  - e.g., mice can prevent cat from eating
- Since cats/mice will eventually finish eating, won't starve forever