CSE120
Principles of Operating Systems

Prof. Yuanyuan (YY) Zhou
Synchronization Review
Synchronization

- Why do you need synchronization?
  - Data race: Multiple threads/processes try to read/write the same shared data
- When do you need synchronization?
  - Accessing shared data
- What is shared data?
- How to synchronize?
  - Critical section
- What is needed to guarantee critical sections
  - Four conditions
A real life example for Synchronization

- Roommate A comes back, checks the refrigerator. If there is no milk, he then goes out to buy milk
- Roommate B also does the same thing
- If the two are not synchronized, what’s the result?

- How can you solve this problem?
Review: Critical Section

Process {
    while (true) {
        ENTER CRITICAL SECTION
        Access shared variables; // Critical Section;
        LEAVE CRITICAL SECTION
        Do other work
    }
}

Which four conditions?
Initially x=0
x is a global variable

Thread 1

int i; //private
for (i=0; i<3; i++)
  x++;

Thread 2

int j; //private
for (j=0; j<3; j++)
  x++;
Implementing Locks (1)

- How do we implement locks?
- Here is one attempt:

```c
struct lock {
    int held = 0;
}
void acquire (lock) {
    while (lock->held);
    lock->held = 1;
}
void release (lock) {
    lock->held = 0;
}
```

- This is called a spinlock because a thread spins waiting for the lock to be released
- Does this work?
Using Test-And-Set

Here is our lock implementation with test-and-set:

```c
struct lock {
    int held = 0;
};
void acquire (lock) {
    while (test-and-set(&lock->held));
}
void release (lock) {
    lock->held = 0;
}
```

- When will the while return? What is the value of held?
- Does it work?
- Does it work on multiprocessors?
Other Similar Hardware Instruction

- Swap = TSL

```c
void Swap (char* x,* y);
\ All done atomically
{
    char temp = *x;
    *x = *y;
    *y = temp
}
```
Use Swap to implement lock acquire and release

```c
struct lock {
    int held = 0;
};

void acquire (lock) {
    Fill in code here
}

void release (lock) {
    Fill in code here
}
```