EECE.4810/EECE.5730: Operating Systems

Spring 2018

Lecture 10: Key Questions
February 28, 2018

1. (Review) Describe semaphores, including the defined operations, the two general types, and how they can be used for both mutual exclusion and ordering.

2. Explain deadlock between threads, using the dining philosophers problem (pseudo-code given below) as an example.

Dining Philosophers: structure of philosopher i:

```
do {
   down(chopstick[i] );
   down(chopStick[ (i + 1) % 5] );

   //  eat

   up(chopstick[i] );
   up(chopstick[ (i + 1) % 5] );

    //  think
} while (TRUE);
```

3. What conditions are necessary for deadlock to occur?

4. What general strategies can be used to handle deadlock?

5. How can deadlock be prevented?

6. Explain the Banker's Algorithm for deadlock prevention.

7. **Example:** Consider a multithreaded bank software package in which the following function is used to transfer money. Assume that the program contains a global array of locks, `locks[]`, with one entry per account number, such that `locks[i]` is the lock for account number `i`.

```
void transfer_money(int src_acct, int dest_acct, int amt) {
     locks[src_acct].lock();   // Lock account sending money
     locks[dest_acct].lock();  // Lock account receiving money
     <transfer money>
     locks[dest_acct].unlock();
     locks[src_acct].unlock();
}
```

Explain how this function can deadlock if called in multiple threads, and rewrite (in pseudo-code or actual code) the function to remove the deadlock condition.