CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Monitors and Dining Philosophers

Monitors can make solution to Dining Philosophers problem easier to write

One of the key issues of the problem is that philosophers can pick up just one chopstick at a time, which can lead to deadlock

We will use monitors to ensure that philosophers only grab chopsticks if both are available

This could be done without monitors (after all -- monitors themselves are implemented on other primitives), but monitors can make our lives easier

Assume we have five philosophers

In addition to tracking whether philosophers are eating or thinking, we keep track of which philosophers would *like* to be eating

```
enum { THINKING, HUNGRY, EATING } state[5];
```

Philosopher can only be eating if both neighbors are not eating

Recall the condition variables we can use to wait for certain criteria to be met

In this case, a philosopher will `wait` if they are hungry but cannot obtain both chopsticks

```
condition self[5];
```

Using `DiningPhilosophers` monitor to control access to chopsticks, solution for philosopher `i` is simply

```
while (true) {
    DiningPhilosophers.pickup(i)
    // eat
    DiningPhilosophers.putdown(i)
}
```

```
monitor DiningPhilosophers
{
    enum { THINKING, HUNGRY, EATING } state[5];
    condition self[5];

    initialization code() {
        for (int i = 0; i < 5; i++)
            state[i] = THINKING;
    }

    // ... other functions
}
```

```
monitor DiningPhilosophers
{
    // variables and initializations

    void pickup(int i) {
        state[i] = HUNGRY;
        test(i);
        if (state[i] != EATING)
            self[i].wait();
    }

    void putdown(int i) {
        state[i] = THINKING;
        test((i + 4) % 5);
        test((i + 1) % 5);
    }

    void test(int i) {
        if ((state[(i + 4) % 5] != EATING) &&
          (state[i] == HUNGRY) &&
          (state[(i + 1) % 5] != EATING)) {
            state[i] = EATING;
            self[i].signal();
        }
    }
}
```

Caution: this solution is not perfect. A philosopher can still starve to death