2020春

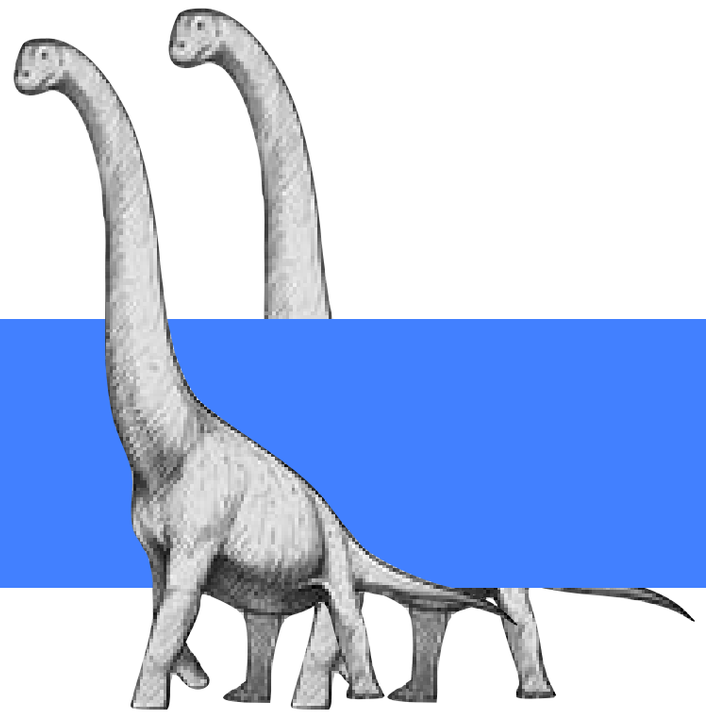
# LINUX OPERATING SYSTEM

YANG

LINUX操作系统（双语）

# 双语课→课件内容中英混排

Lecture 10

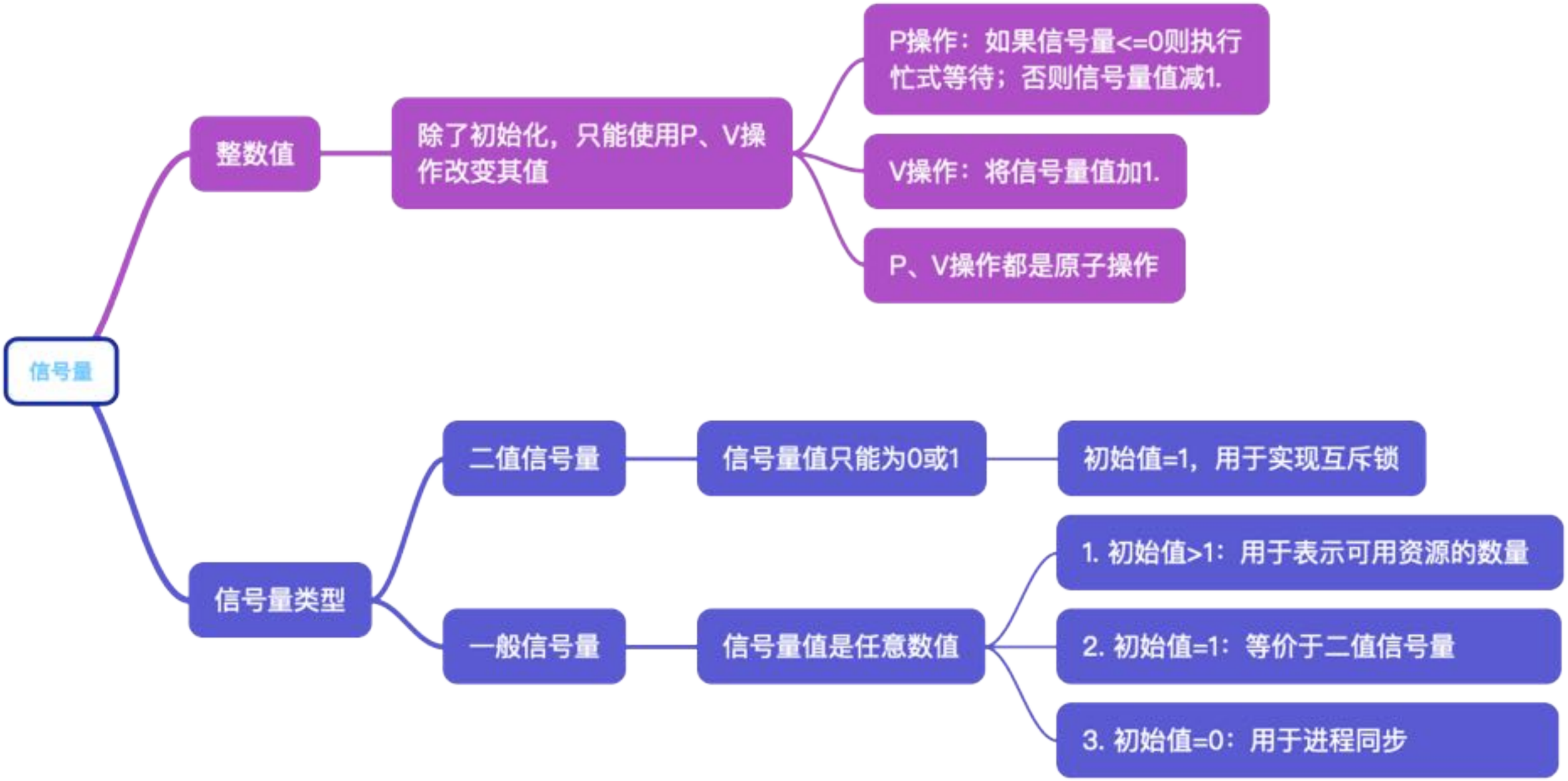


# Semaphores II

# 本讲内容

- 信号量实现同步
- 经典同步问题
- 生产-消费者问题
- 苹果桔子问题

# REVIEW


信号量
整数值
除了初始化，只能使用P、V操作改变其值
P操作：如果信号量<=0则执行忙式等待；否则信号量值减1.
V操作：将信号量值加1.
P、V操作都是原子操作
信号量类型
二值信号量
信号量值只能为0或1
初始值=1，用于实现互斥锁
一般信号量
信号量值是任意数值
1. 初始值>1：用于表示可用资源的数量
2. 初始值=1：等价于二值信号量
3. 初始值=0：用于进程同步

# 信号量实现同步

# 司机与售票员

- 演员
  - 司机：启动车辆；正常行车；到站停车
  - 售票员：关车门；售票；开车门
- 规则
  - 司机要等车门关闭才能开车
  - 售票员要等车停下才能开门

# 信号量实现同步

| 司机 | 售票员 |
| --- | --- |
| 启动车辆 | 关车门 |
| 正常行车 | 售票 |
| 到站停车 | 开车门 |

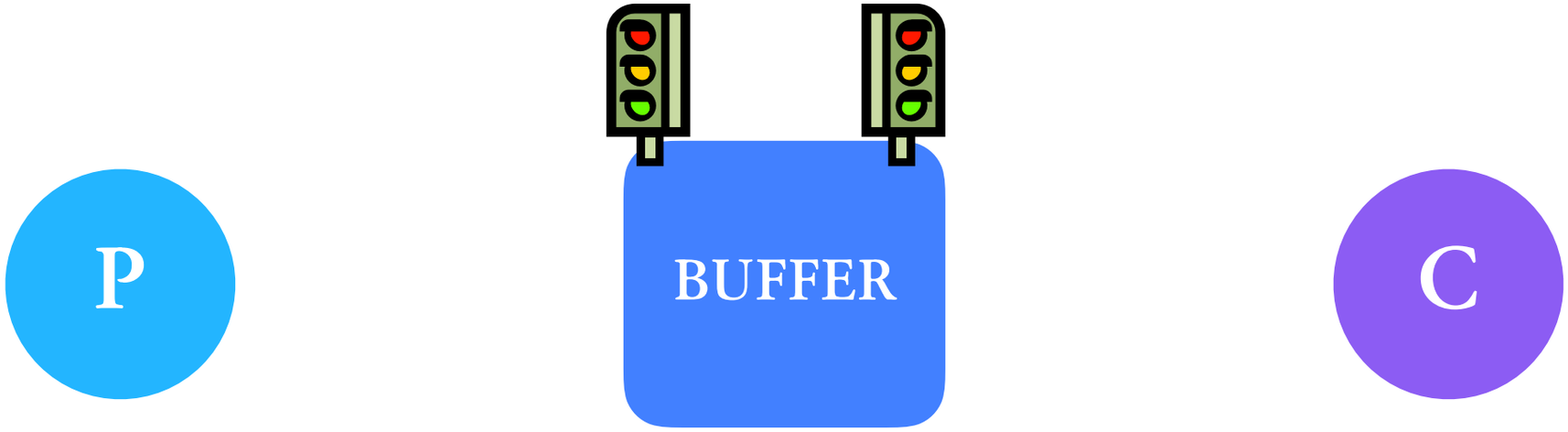
# 同步问题

- 同步问题实质是将异步的并发进程按照某种顺序执行；
- 解决同步的本质就是要找到并发进程的交互点，利用P操作的等待特点来调节进程的执行速度；
- 通常初始值为0的信号量可以让进程直接进行等待状态，直到另一个进程唤醒他。

# 经典同步问题

# 生产-消费者问题

- 生产者(P)与消费者(C)共用一个缓冲区，生产者不能往“满”的缓冲区中放产品，消费者不能从“空”的缓冲区中取产品。



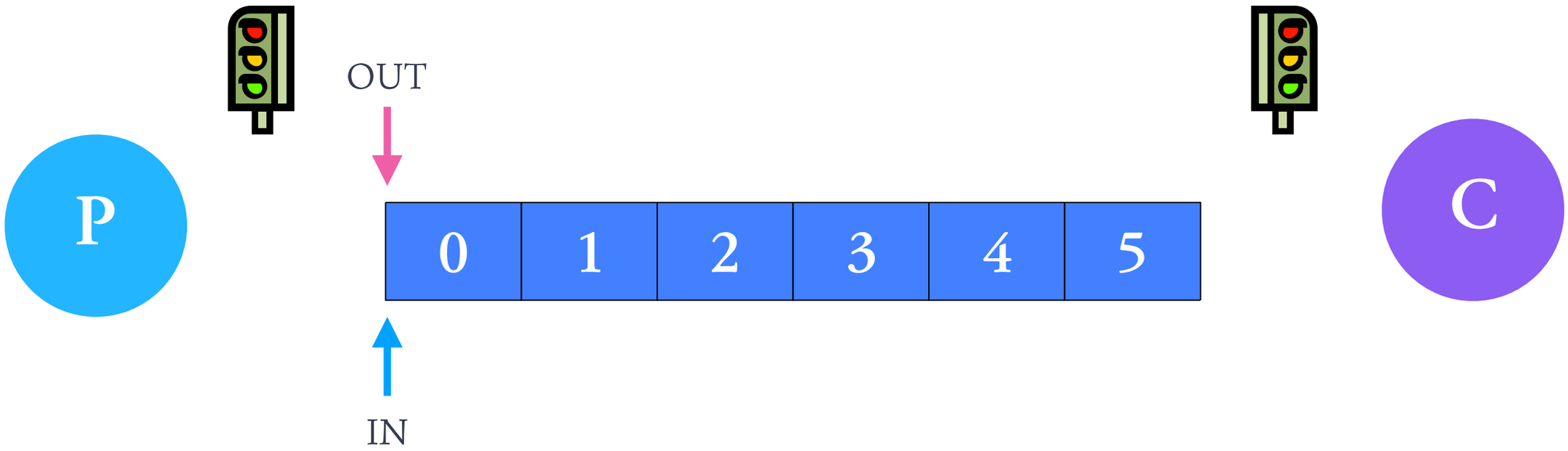
# 单缓冲解决方案

```
Semaphore empty = 1;     //signal for producer
Semaphore full  = 0;     //signal for consumer
```

```
Producer {
    while (true) {
        make a product;
        P(empty);
        put the product into buffer;
        V(full);
    }
}
```

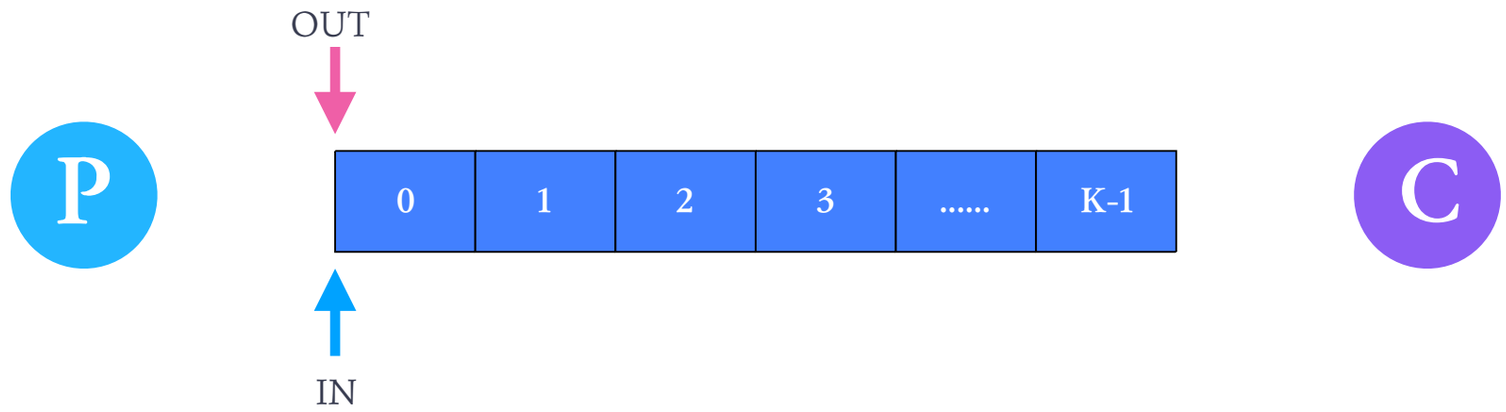
```
Consumer {
    while (true) {
        P(full);
        pick product from buffer;
        V(empty);
        consume the product;
    }
}
```

# THE BOUNDED-BUFFER PROBLEM



# THE BOUNDED-BUFFER PROBLEM

```
item B[k];
semaphore empty = k;
semaphore full = 0;
int in = 0, out = 0;
```




```
Process producer_i {
    make a product;
    P(empty);

    B[in] = product;
    in = (in+1) % k;

    V(full);
}
```

```
Process consumer_i {
    P(full);

    product = B[out];
    out = (out+1) % k;

    V(empty);
    consume a product;
}
```

# THE BOUNDED-BUFFER PROBLEM

```
item B[k];
semaphore empty = k;
semaphore full = 0;
int in = 0, out = 0;
semaphore mutex = 1;

Process producer_i {
    make a product;
    P(empty);
    P(mutex);
    B[in] = product;
    in = (in+1) % k;
    V(mutex);
    V(full);
}
```

```
Process consumer_i {
    P(full);
    P(mutex);
    product = B[out];
    out = (out+1) % k;
    V(mutex);
    V(empty);
    consume a product;
}
```

# 苹果桔子问题

- 问题描述
  - 桌上有一只盘子，每次只能放入一只水果
  - 爸爸专向盘子中放苹果，妈妈专向盘子中放桔子
  - 儿子专等吃盘子中的桔子，女儿专等吃盘子里的苹果

# 苹果桔子解决方案

```
semaphore sp     = 1;   /* 盘子里允许放一个水果*/
semaphore sg1    = 0;   /* 盘子里没有桔子 */
semaphore sg2    = 0;   /* 盘子里没有苹果*/
```

```
Process father  {
    削一个苹果;
    P(sp);
    把苹果放入plate;
    V(sg2);
}
```

```
Process daughter {
    P(sg2);
    从plate中取苹果;
    V(sp);
    吃苹果;
}
```

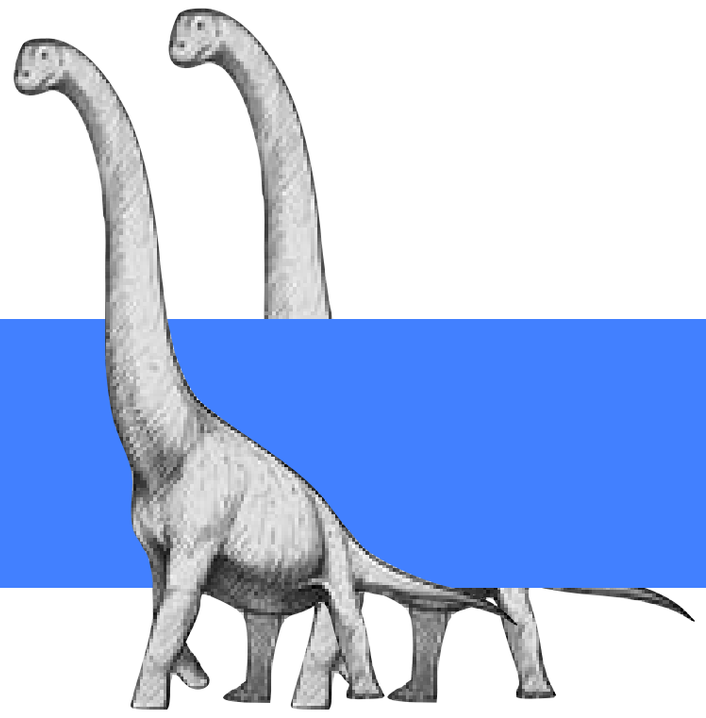
```
Process mother {
    剥一个桔子;
    P(sp);
    把桔子放入plate;
    V(sg1);
}
```

```
Process son {
    P(sg1);
    从plate中取桔子;
    V(sp);
    吃桔子;
}
```

# 下期预告

- 下次直播时间：3月9日 上午9:30
- 课程内容
  - Practice 5 Pthread Synchronization Applications
- 作业2
  - 仔细阅读并理解教材的“读者-写者同步问题”；
  - 你们已经具备了Pthread同步工具的使用能力！将今日课堂讲的3个同步问题编程运行起来，观察结果！
  - 选择1份代码+结果截图（存成pdf，文件名同前）于周日晚23:59前提交至群文件里！

Lecture 10



# The End