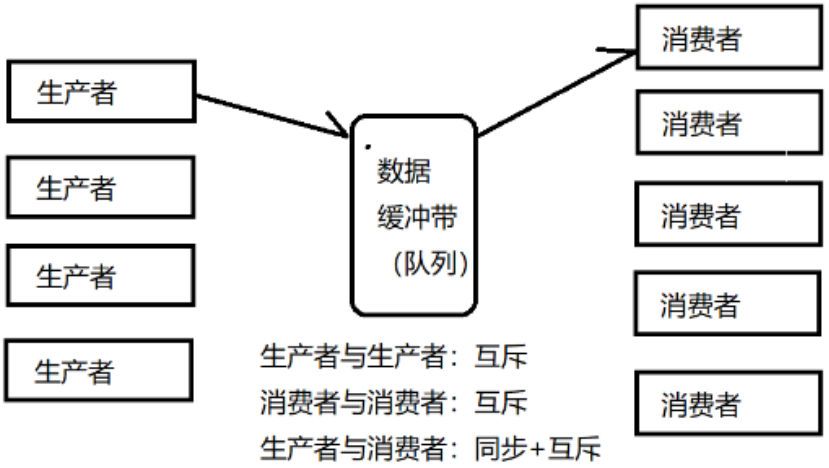
生产者与消费者模型:一种设计模式---大佬们针对典型场景设计的解决方案



生产者与消费者模型所解决的问题：解耦和，支持并发，支持忙闲不均

生产者与消费者模型的实现:
生产者与消费者只是不同角色的执行流;
只需要中间实现线程安全的队列，然后再各自创建不同角色的执行流就可以实现这个模型
线程安全的阻塞队列的实现:
使用c++封装实现一个阻塞队列类:: stl中我们学习了队列容器std::queue--但是这个队列是非线程安全的

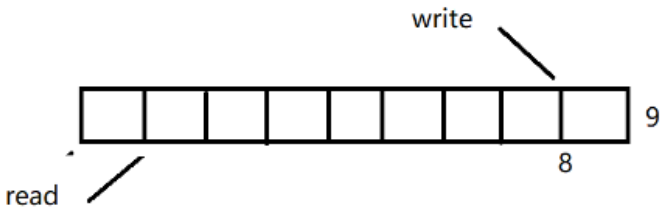
```
class BlockQueue{
private:
    std::queue<int>_ queue; //stl中queue
    int_ capcity; //定义队列中最大的节点数量
    pthread_ mutex _t_ mutex; //互斥锁保护临界资源queue的访问
    pthread_ cond. _t _ cond_ con;//消费者等待队列，实现同步
    pthread_ cond _t _ cond_ pro://生产 者等待队列，实现同步
public:
    BlockQueue(int max _que = MAX_ QUEE).初始化过程..}
    ~ BlockQueue () {..资源销毁过程..}
    bool Push(int data);//生产者入队操作
    bool Pop(int *data);//消费者出队操作
}
```

生产者与消费者模型:
1.这个模型的应用场景:
2.这个模型针对这种场景解决了什么问题:
3.生产者与消费者模型的实现流程

POSIX标准信号量:千万不要跟信号概念混淆了
作用:实现进程/线程间的同步与互斥
本质:一个计数器+ pcb等待队列
实现同步:
计数器对资源数量进行计数，当线程想要获取资源的时候，先访问信号量,判断是否能够获取(信号量通过自身的计数完成判断) ,若计数< =0则直接阻塞线程，计数-1;其它线程生产资源之后，计数+1,唤醒等待队列上的pcb
实现互斥:
保证计数器的数值不会大于1,就表示同一时间只有一个线程能够访问资源;

操作接口:
1.定义信号量sem_t sem;
2.初始化信号量int sem_ init(sem_t *sem, int pshared, int value)
pshared:信号量即可用于进程间也可用于线程间，pshared为0表示用于线程间　非0表示用于进程间
value:信号量就是一个计数器，统计资源数量，value就是通过资源数量初始化计数器的
3.在访问临界资源之前，先判断计数，是否能够访问资源，若不能访问，则阻塞线程;若可以访问则调用直接返回
int sem_wait(sem _t *sem) / int sem_ trywait(sem_t *sem)/ int sem_timedwait(sem _t *sem, struct timespec *ts)
4.访问临界资源之后/生产资源之后，唤醒一个等待的线程，并且计数+ 1
int sem_ post(sem_ t *sem);
5.不使用信号量记得释放资源
int sem_destroy(sem_t *sem);

信号量的应用:
通过信号量实现一个环形队列，最终实现一个生产者与消费者模型



read == write ， 队列为空

write +1 % capacity == read　队列中数据满了

class RingQueue{
private:
　　std:vector <int>_ queue(node_ num);
　　int_ step_read; //当前即将读取数据的位置的下标
　　int_ step_ write; //当前即将写入数据的位置的下标
　　sem _t_sem_lock;//用于实现互斥的信号量
　　sem_ t_sem_ data; //使用这个计数器,实现对当前队列中的数据资源的数量进行计数;如果< =0表示没有资源，则消费者会陷入等待
　　sem_ t_ sem_ space; //使用这个计数器，实现对当前队列中的空闲空间数量进行计数;如果< =0表示队列满了，则生产者陷入等地啊
public:
　　RingQueue (int max = MAX_ QUEUE) {..初始化过程.}
　　~ RingQueue(){...销毁资源过程..}
　　bool Push(int data);
　　bool Pop(int *data);
}
信号量:
1.信号量的本质
2.信号量的作用
3.信号量实现互斥的原理
4.信号量实现同步的原理
5.信号量实现环形队列的流程实现

复习：

多线程:
生产者与消费者模型:
典型场景:有生产者生产数据，消费者获取数据进行处理的场景
一个场所， 两种角色，三种关系
生产者与消费者模型针对这种场最所解决的问题:
1.解耦和 2. 支持并发 3. 支持忙闲不均
实现:
生产者与消费者就是两种不同业务的执行流
需要更多实现的是中间的一个场所----线程安全的队列

```
class BlockQueue {
private:
    std:queue <int>_ queue;
    int_ capacity;
   pthread_ mutex_t_ mutex;
   pthread_ cond_t_cond_consumer;
   pthread_ cond_t_cond_productor;
}
```

信号量:
1.本质就是一个计数器+ pcb等待队列;　　突出的重 点是什么
2.可以实现线程/进程间的同步与互斥　　解释为什么是这样的
3.同步--通过条件判断实现对临界资源访问的合理性
　互斥--通过唯一-访问实现对临界资源访问的安全性

同步的实现:
信号量通过自身的计数器对资源进行计数，实现在计数>0的时候，执行流能够访问临界资源;否则在执行流访问前则是执行流阻塞;等到其它执行流生产资源之后，计数器+1,然后唤醒等待的执行流。
互斥的实现:
信号量通过自身计数器，在计数最大不大于1的情况下，表示同一时间只有一个执行流能够访问资源，实现互斥
信号量的操作流程:
1.定义信号量变量sem_ t sem;
2.初始化信号量int sem_init(sem_t *sem, int pshared, int value)
3.在访问临界资源前先访问信号量，判断是否能够访问
int sem _wait(sem. t *sem); / sem_timedwait()/ sem_trywait()
4.其它执行流产生资源之后，计数+1,唤醒等待的执行流int sem_post(sem _t *sem);
5.释放资源int sem_destroy(sem_t *sem)

线程安全的时候:通过同步与互斥实现的
同步的实现方式:条件变量和信号量
互斥的实现方式:互斥锁和信号量

条件变量和信号量有什么区别?

1.条件变量促使线程等待的条件判断需要用户自身完成，而信号量是通过自身计数进行判断的

2.条件变量需要搭配互斥锁一起使用，但是信号量不需要

信号量与互斥锁有什么区别?

1.信号量是对资源进行计数的计数器，主要是用于实现同步的:而互斥锁只能实现互斥