C++11 线程库提供的条件变量condition\_variable，就是Linux平台下的Condition Variable机制，用于解决线程间的同步通信问题，下面通过代码演示一个生产者-消费者线程模型，仔细分析代码：

#include <iostream> // std::cout

#include <thread> // std::thread

#include <mutex> // std::mutex, std::unique\_lock

#include <condition\_variable> // std::condition\_variable

#include <vector>

// 定义互斥锁(条件变量需要和互斥锁一起使用)

std::mutex mtx;

// 定义条件变量(用来做线程间的同步通信)

std::condition\_variable cv;

// 定义vector容器，作为生产者和消费者共享的容器

std::vector<int> vec;

// 生产者线程函数

void producer()

{

// 生产者每生产一个，就通知消费者消费一个

for (int i = 1; i <= 10; ++i)

{

// 获取mtx互斥锁资源

std::unique\_lock<std::mutex> lock(mtx);

// 如果容器不为空，代表还有产品未消费，等待消费者线程消费完，再生产

while (!vec.empty())

{

// 判断容器不为空，进入等待条件变量的状态，释放mtx锁，

// 让消费者线程抢到锁能够去消费产品

cv.wait(lock);

}

vec.push\_back(i); // 表示生产者生产的产品序号i

std::cout << "producer生产产品:" << i << std::endl;

/\*

生产者线程生产完产品，通知等待在cv条件变量上的消费者线程，

可以开始消费产品了，然后释放锁mtx

\*/

cv.notify\_all();

// 生产一个产品，睡眠100ms

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

}

}

// 消费者线程函数

void consumer()

{

// 消费者每消费一个，就通知生产者生产一个

for (int i = 1; i <= 10; ++i)

{

// 获取mtx互斥锁资源

std::unique\_lock<std::mutex> lock(mtx);

// 如果容器为空，代表还有没有产品可消费，等待生产者生产，再消费

while (vec.empty())

{

// 判断容器为空，进入等待条件变量的状态，释放mtx锁，

// 让生产者线程抢到锁能够去生产产品

cv.wait(lock);

}

int data = vec.back(); // 表示消费者消费的产品序号i

vec.pop\_back();

std::cout << "consumer消费产品:" << data << std::endl;

/\*

消费者消费完产品，通知等待在cv条件变量上的生产者线程，

可以开始生产产品了，然后释放锁mtx

\*/

cv.notify\_all();

// 消费一个产品，睡眠100ms

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

}

}

int main()

{

// 创建生产者和消费者线程

std::thread t1(producer);

std::thread t2(consumer);

// main主线程等待所有子线程执行完

t1.join();

t2.join();

return 0;

}

代码运行结果如下，可以看到，生产者和消费者线程交替生产产品和消费产品，两个线程之间进行了完美的通信协调运行。

producer生产产品:1

consumer消费产品:1

producer生产产品:2

consumer消费产品:2

producer生产产品:3

consumer消费产品:3

producer生产产品:4

consumer消费产品:4

producer生产产品:5

consumer消费产品:5

producer生产产品:6

consumer消费产品:6

producer生产产品:7

consumer消费产品:7

producer生产产品:8

consumer消费产品:8

producer生产产品:9

consumer消费产品:9

producer生产产品:10

consumer消费产品:10