**Project 5:**

**Designing a Thread Pool**

**& Producer-Consumer Problem**

Project 5-1: Designing a Thread Pool

课本中提供了两种可供选择的语言——C或者java。我选择使用C语言来完成该项目。

（一）问题分析

我需要补充threadpool.c源文件中的几个函数，以及一些其他必要的功能。

（二）实现细节

一、全局变量

1、task\_queue[]：任务列表，用数组实现

2、worktodo：当前任务

3、threadpool[]：线程池

4、mutex：任务进出任务列表时使用的互斥锁

5、semaphore：任务执行时选择线程使用的信号量

6、curLEN：当前任务列表长度

7、thread\_working[]：线程池中线程的占用情况

二、函数

1、pool\_init()

初始化全局变量：当前任务列表长度为0，线程池中所有线程未被占用，初始化互斥锁和信号量。

2、enqueue()

首先如果任务列表已满，则无法再有新任务进入；如果任务列表未满，需要用互斥锁保证同一时间只有一个任务进入列表并修改curLEN，保证任务列表中的实际任务个数与curLEN一致。

3、dequeue()

同理，如果任务列表为空，则没有任务能被执行；如果有任务列表中有任务还未执行，需要用互斥锁保证同一时间只有一个任务被挪出任务列表并curLEN，保证任务列表中的实际任务个数与curLEN一致。

4、pool\_submit()

（1）向线程池提交任务，如果任务列表未满，将该任务加入任务列表。

（2）为任务列表的第一个任务，即下一个要执行的任务分配线程，使用信号量保证同一时间只有一个线程被分配给一个任务，保证一个线程同一时间只执行一个任务，并修改该线程的占用情况。

5、\*work()

每一个thread在join后执行的函数，函数参数已传入执行任务的线程编号，执行完该任务后，修改该线程的占用情况，释放信号量，退出线程。

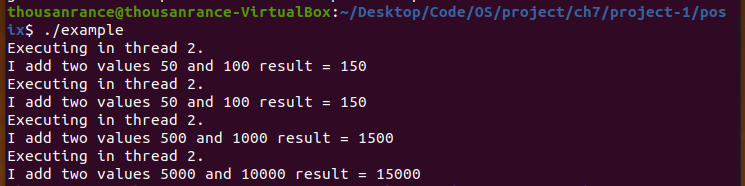
6、pool\_shutdown()

执行完线程池中的所有线程，最后删除互斥锁和信号量。

三、主函数

多添加几个work进行实验

（三）运行结果



Project 5-2: Producer-Consumer Problem

课本中提供了两种可供选择的API——Pthreads或者Windows API。我选择使用Pthreads来完成该项目。

（一）问题分析

根据课本给出的Outline of buffer operation，补充完成必要的功能，需要完善buffer的功能、producer和consumer线程的执行函数和主函数。

为了简化Makefile，我将除了buffer.h外的代码写在一个源文件producer\_consumer.c里。

（二）实现细节

一、buffer

1、全局变量

buffer[BUFFER\_SIZE]：缓冲区，用数组实现列表

curSize：记录当前buffer中有多少item

2、insert\_item()

如果缓冲区未满，将其添加到列表最后。

3、remove\_item()

如果缓冲区不为空，将第一个item移除。

二、main()

1. Get command line arguments argv[1],argv[2],argv[3]

首先检查输入是否合法，合法的话，存下这些参数。

sleep\_time <- argv[1]: How long to sleep before terminating

num\_of\_producer <- argv[2]: The number of producer threads

num\_of\_consumer <- argv[3]: The number of consumer threads

2. Initialize buffer

curSize置零。初始化互斥锁和信号量。

Producer-Consumer问题需要三个信号量，mutex、empty、full。

mutex提供缓冲区访问的互斥要求，初始化为1；empty和full分别用于表示空的和满的缓冲区空间的数量，初始化为n和0。

3. Create producer thread(s)

使用pthread\_create(),按照输入的参数num\_of\_producer创建线程。

4. Create consumer thread(s)

使用pthread\_create(),按照输入的参数num\_of\_consumer创建线程。

5. Sleep

使用sleep(),参数为输入的sleep\_time。

6. Exit

使用pthread\_cancel()停止所有线程。

删除互斥锁和信号量。

三、producer and consumer threads

1、\*producer()

首先等待随机的时间。

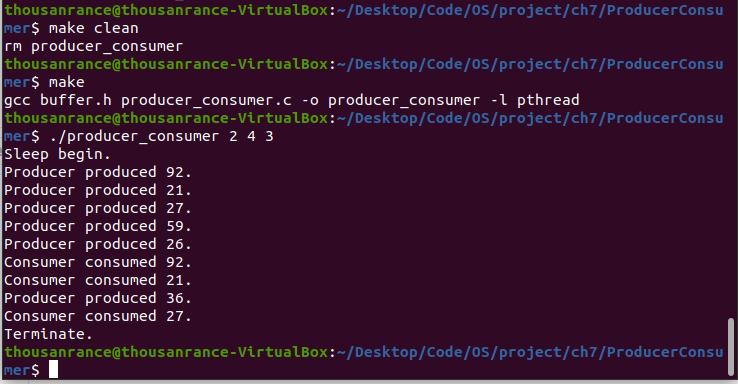
生产者生产item，然后将其加入到buffer中。对于生产者，empty信号量功能为在buffer不满时允许生产者将item加入到buffer中；full信号量的功能为告诉消费者有item可以消耗；使用mutex锁住insert的过程，保证生产者和消费者在同一时间只有一个能够访问缓冲区。

2、\*consumer()

首先等待随机的时间。

消费者item，将其加入从buffer中移除。对于消费者，full信号量功能为在buffer不空时允许消费者消耗buffer中的item；empty信号量的功能为告诉生产者buffe中有空位可以填入item；使用mutex锁住remove的过程，保证生产者和消费者在同一时间只有一个能够访问缓冲区。

（三）运行结果



注：由于运行时间是有限的，所以在运行结束时，consumer有可能没有消耗完producer生产的item。