**PingCAP小作业概要说明**

# 一、需求分析

根据小作业的要求实现一个任务队列，生产者线程可以向任务队列中添加任务，消费者线程可以从任务队列中取出任务执行，执行完成后从任务队列中删除。并且需要保证并发安全，并且出现异常时可以做出关闭队列的操作。

所以可以得出基本要求是实现一个任务队列，可以进行插入删除任务操作，并且满足其他的一些特性。

# 二、方案设计

根据需求分析，任务队列的设计大致包括数据结构、接口设计、并发安全、功能划分、出错处理等几个方面。

1. 首先是数据结构，因为任务队列需要进行插入删除查询操作，所以选择了map（红黑树）作为底层的数据结构，这样可以做到插入删除查询都具有较高的效率。
2. 接口设计。根据需求的添加执行删除要求，所以需要add、get、done接口；并且需求中需要关闭队列，索引需要shutdown、is\_closed等接口；并且其他的通用接口length等。
3. 并发安全。对于并发安全，有以下几种方式。一、采用原子操作的模式，这种操作方式的优点是效率高，但是它在关掉任务队列时的处理比较复杂；二、采用锁的模式，这种操作方式的优点是关掉任务队列时的处理更好控制，缺点是效率低。为了关掉任务队列时的处理，所以选择了第二种方式。
4. 功能划分。为了各个模块的功能比较清晰，所以选择了多级queue，第一级queue只是简单的进行插入删除查询操作，不进行并发控制。第二级queue在第一级queue的基础上实现并发控制，并且进行对应用暴露任务队列的接口。锁则由专门的一个模块来实现。

# 三、代码结构

1、文件构成

代码主要有以下几个文件组成：

以下文件是任务队列的实现：

PingCAP\_cond.h

PingCAP\_define.h

PingCAP\_mutex.h

PingCAP\_simple\_queue.h

PingCAP\_task\_item.h

PingCAP\_task\_queue.h

以下文件是测试任务队列是否满足要求编写的测试代码

PingCAP\_test\_task\_queue.h

main.cpp

对于任务队列的相关文件的说明：

PingCAP\_define.h：相关变量的定义以及一些宏定义

PingCAP\_mutex.h：互斥锁简单封装

PingCAP\_cond.h：互斥锁条件变量简单封装

PingCAP\_task\_item.h：对于用户任务的封装，添加了task\_id,以便于在任务队列中存储

PingCAP\_simple\_queue.h：任务队列的存储结构，可以进行插入删除查询操作，不保证并发安全

PingCAP\_task\_queue.h：根据要求最终实现的任务队列

2、类构成

PingCAPMutex：互斥锁简单封装

PingCAPCond：继承自PingCAPMutex互斥锁条件变量简单封装

template <typename T>

TaskItem：对用户任务的封装，用户任务类为参数类T

template <typename T>

SimpleQueue：简单的队列，可以进行插入删除查询操作，不保证并发安全，内部包含了一个stl的容器map类的成员变量。接口主要有

int size() const;

void clear();

bool push(TaskItem<T> \*&task);

bool get(TaskItem<T> \*&task);

bool pop(TaskItem<T> \*&task);

template <typename T>

TaskQueue：根据要求最终实现的任务队列，内部包含了一个SimpleQueue类的成员变量。接口主要有

int length();

bool add(TaskItem<T> \*task, int maxQueueLen = 0, bool block = true); bool get(TaskItem<T> \*&task, bool wait);

bool done(TaskItem<T> \*&task);

bool is\_closed();

# 四、致歉

由于我现在还在学校实验室，老师要求之前做的项目还在一直推进，再加上学校80周年校庆，所以导致最近时间不太充足，很多东西没有做好，我在这里表示歉意。不过后续会继续完善。对方案进行进一步的优化、完善测试以及完成代码的UML类图。