[原]linux下c++实现简单的生产者消费者队列模式

2014-4-23 阅读407 评论0

引言

生产者消费者是一个经典的模式

利用生产者，消费者和缓冲区降低了生产者和消费者之间的的耦合度

便于对生产者和消费者的修改

下面记录的是一个经典的单一生产者多消费者的模式

设计思路

以队列做为缓冲区，实现产品的FIFO

生产者调用缓冲区的push函数，将产品加入缓冲区

消费者调用缓冲区的pop函数，将产品从缓冲区取出

因为生产者与消费者分属于不同的线程，所以要设置锁

类的声明

```
class CacheQueue
{
    private:
        /**
         * @brief 缓冲队列
         */
        queue<int>* _requests;

        /**
         * @brief 互斥锁
         **/
        pthread_mutex_t _mutex;

        /**
         * @brief Queue not full conditional object
         **/
        pthread_cond_t _not_full_cond;

        /**
         * @brief Queue not empty conditional object
         **/
        pthread_cond_t _not_empty_cond;

        uint32_t _bufSize;

    public:

        ChacheQueue();

        void SetMaxLength(uint32_t bufSize);
        /**
          * @brief 向队列添加产品
          * @param [in] req: 待添加的产品
          **/
```

```
        void Push(int req);

        /**
          * @brief 从队列中取出一个产品
          * @param [return] : 从队列中取出的产品
          **/
        int Pop(uint32_t timeout);

        /**
          * @brief 析构函数
          **/
        ~CacheQueue();
};
```

重要的函数是Push和Pop，生产者调用Push向缓冲区添加产品，消费者则调用Pop函数获取产品

线程条件_not_full_cond表示队列不满，可以添加产品

线程条件_not_empty_cond表示队列不空，可以获取产品

Push函数

```
void CacheQueue::Push(int req)
{
        /**
        * 上锁
        */
    pthread_mutex_lock(&_mutex);

        /**
        * 如果队列满，等待信号
        */
    while (_requests->size() == _bufSize)
    {
        pthread_cond_wait(&_not_full_cond, &_mutex);
    }
    _requests->push(req);

        /**
        * 发送非空信号
        */
    pthread_cond_signal(&_not_empty_cond);

        /**
        * 解锁
        */
    pthread_mutex_unlock(&_mutex);
}
```

Pop函数

```
int CacheQueue::Pop(uint32_t timeout)
{
    int ret = 0;
    int req = NO_DATA;
        /**
        * 上锁
        */
    pthread_mutex_lock(&_mutex);
```

```
    /**
    * 若队列空等待指定时间
    */
struct timeval now;
    struct timespec timepass;
    gettimeofday(&now, NULL);
    timepass.tv_sec = now.tv_sec + timeout;
    timepass.tv_nsec = 0;
while (ret == 0 && _requests->empty())
{
            ret = pthread_cond_timedwait(&_not_empty_cond, &_mutex, &timepass);
}
    /**
    * 没有数据，返回没有数据标识
    */
if(ret!=0)
{
    pthread_mutex_unlock(&_mutex);
    return req;
}
    /**
    * 返回数据，发送队列非满信号
    */
req = _requests->front();
_requests->pop();
pthread_cond_signal(&_not_full_cond);
    /**
    * 解锁
    */
pthread_mutex_unlock(&_mutex);
    return req;
}
```

上一篇

请先登录后，再发表评论！

查看评论

更多评论（0）

回顶部