最新版Web服务器项目详解 - 01 线程同步机制封装类

原创 互联网猿 两猿社 2020-04-21 14:30

基础知识

RAII

- RAII全称是“Resource Acquisition is Initialization”，直译过来是“资源获取即初始化”.
- 在构造函数中申请分配资源，在析构函数中释放资源。因为C++的语言机制保证了，当一个对象创建的时候，自动调用构造函数，当对象超出作用域的时候会自动调用析构函数。所以，在RAII的指导下，我们应该使用类来管理资源，将资源和对象的生命周期绑定
- RAII的核心思想是将资源或者状态与对象的生命周期绑定，通过C++的语言机制，实现资源和状态的安全管理,智能指针是RAII最好的例子

信号量

信号量是一种特殊的变量，它只能取自然数值并且只支持两种操作：等待(P)和信号(V).假设有信号量SV，对其的P、V操作如下：

> - P，如果SV的值大于0，则将其减一；若SV的值为0，则挂起执行
> - V，如果有其他进行因为等待SV而挂起，则唤醒；若没有，则将SV值加一

信号量的取值可以是任何自然数，最常用的，最简单的信号量是二进制信号量，只有0和1两个值.

> - sem_init函数用于初始化一个未命名的信号量
> - sem_destory函数用于销毁信号量

- sem_wait函数将以原子操作方式将信号量减一,信号量为0时,sem_wait阻塞
- sem_post函数以原子操作方式将信号量加一,信号量大于0时,唤醒调用sem_post的线程

以上，成功返回0，失败返回errno

互斥量

互斥锁,也成互斥量,可以保护关键代码段,以确保独占式访问.当进入关键代码段,获得互斥锁将其加锁;离开关键代码段,唤醒等待该互斥锁的线程.

- pthread_mutex_init函数用于初始化互斥锁
- pthread_mutex_destory函数用于销毁互斥锁
- pthread_mutex_lock函数以原子操作方式给互斥锁加锁
- pthread_mutex_unlock函数以原子操作方式给互斥锁解锁

以上，成功返回0，失败返回errno

条件变量

条件变量提供了一种线程间的通知机制,当某个共享数据达到某个值时,唤醒等待这个共享数据的线程.

- pthread_cond_init函数用于初始化条件变量
- pthread_cond_destory函数销毁条件变量
- pthread_cond_broadcast函数以广播的方式唤醒**所有**等待目标条件变量的线程
- pthread_cond_wait函数用于等待目标条件变量.该函数调用时需要传入 **mutex参数(加锁的互斥锁)** ,函数执行时,先把调用线程放入条件变量的请求队列,然后将互斥锁mutex解锁,当函数成功返回为0时,互斥锁会再次被锁上. **也就是说函数内部会有一次解锁和加锁操作**.

功能

锁机制的功能

- 实现多线程同步，通过锁机制，确保任一时刻只能有一个线程能进入关键代码段.

封装的功能

- 类中主要是Linux下三种锁进行封装，将锁的创建于销毁函数封装在类的构造与析构函数中，实现RAII机制

```
1  class sem{
2      public:
3          //构造函数
4          sem()
5          {
```

```
            //信号量初始化
            if(sem_init(&m_sem,0,0)!=0){
                throw std::exception();
            }
        }
        //析构函数
        ~sem()
        {
            //信号量销毁
            sem_destroy(&m_sem);
        }
    private:
        sem_t m_sem;
};
```

- 将重复使用的代码封装为函数，减少代码的重复，使其更简洁

```
    //条件变量的使用机制需要配合锁来使用
    //内部会有一次加锁和解锁
    //封装起来会使得更加简洁
    bool wait()
    {
        int ret=0;
        pthread_mutex_lock(&m_mutex);
        ret=pthread_cond_wait(&m_cond,&m_mutex);
        pthread_mutex_unlock(&m_mutex);
        return ret==0;
    }
    bool signal()
    {
        return pthread_cond_signal(&m_cond)==0;
    }
```

如果本文对你有帮助， 阅读原文 star一下服务器项目，我们需要你的星星^_^.

完。

下一篇 · 最新版Web服务器项目详解 - 02 半同步半反应堆线程池（上）

阅读原文