

C 博客

登录 | 注册

Q

≡

flyingdon的专栏

目录视图

摘要视图

RSS 订阅

个人资料



flyingdon

+ 加关注

发私信

访问：112674次

积分：1098

等级：BLOG > 4

排名：千里之外

原创：5篇

转载：28篇

译文：0篇

评论：11条

文章搜索

Q

文章分类

C语言 (4)

linux内核函数 (2)

linux编程 (5)

linux网络编程 (5)

linux驱动 (15)

不错的博客 (1)

文章存档

2011年08月 (1)

2011年03月 (1)

2011年02月 (1)

2011年01月 (2)

2010年03月 (4)

展开

阅读排行

理解Semaphore及其用法

(24526)

C语言根据日期判断星期

(17461)

Kmalloc

(17053)

解读set_gpio_ctrl(GPIO_

(4937)

fseek()/ftell()/rewind()/fse

(4648)

arm linux下的关于time部

(4640)

评论送书 | 云原生、Docker、Web算法 为什么我们创业失败了和选择创业公司的思考 征文 | 你会为 AI 转型么？

转 理解Semaphore及其用法详解

标签：semaphore thread signal initialization symbian access

2009-12-31 09:35 24550人阅读 评论

快速回复

☆ 我要收藏

分类： linux驱动 (14)

Mutex是一把钥匙，一个人拿了就可进入一个房间，出来的时候把钥匙交给队列的第一个。一般的用法是用于串行化对critical section代码的访问，保证这段代码不会被并行的运行。

Semaphore是一件可以容纳N人的房间，如果人不满就可以进去，如果人满了，就要等待有人出来。对于N=1的情况，称为binary semaphore。一般的用法是，用于限制对于某一资源的同时访问。

Binary semaphore与Mutex的差异：

在有的系统中Binary semaphore与Mutex是没有差异的。在有的系统上，主要的差异是mutex一定要由获得锁的进程来释放。而semaphore可以由其它进程释放（这时的semaphore实际就是个原子的变量，大家可以加或减），因此semaphore可以用于进程间同步。Semaphore的同步功能是所有系统都支持的，而Mutex能否由其他进程释放则未定，因此建议mutex只用于保护critical section。而semaphore则用于保护某变量，或者同步。

关于semaphore和mutex的区别，网上有著名的厕所理论（http://koti.mbnet.fi/niclasw/MutexSemaphore.html）：

Mutex:Is a key to a toilet. One person can have the key - occupy the toilet - at the time. When finished, the person gives (frees) the key to the next person in the queue.Officially: "Mutexes are typically used to serialise access to a section of re-entrant code that cannot be executed concurrently by more than one thread. A mutex object only allows one thread into a controlled section, forcing other threads which attempt to gain access to that section to wait until the first thread has exited from that section."

Ref: Symbian Developer Library(A mutex is really a semaphore with value 1.)

Semaphore:

Is the number of free identical toilet keys. Example, say we have four toilets with identical locks and keys. The semaphore count - the count of keys - is set to 4 at beginning (all four toilets are free), then the count value is decremented as people are coming in. If all toilets are full, ie. there are no free keys left, the semaphore count is 0. Now, when eq. one person leaves the toilet, semaphore is increased to 1 (one free key), and given to the next person in the queue.

Officially: "A semaphore restricts the number of simultaneous users of a shared resource up to a maximum number. Threads can request access to the resource (decrementing the semaphore), and can signal that they have finished using the resource (incrementing the semaphore)."

Ref: Symbian Developer Library

关闭

http://blog.csdn.net/flyingdon/article/details/5110582

1/6

cdev_init()函数	(3184)
Linux驱动程序开发 - 设备	(2951)
一步步设计自己的驱动程	(2617)
kmalloc VS kmem_cach	(2254)

评论排行	
一步步设计自己的驱动程	(4)
解读set_gpio_ctrl(GPIO_	(2)
理解Semaphore及其用法	(2)
Linux驱动程序开发 - 设备	(1)
cdev_init()函数	(1)
linux网络编程基本流程	(1)
C语言根据日期判断星期	(0)
不错的博客	(0)
socket编程中的select (z	(0)
字符设备驱动模板	(0)

推荐文章	
* CSDN日报20170721——《为什么我们创业失败了和选择创业公司的思考》	
* 深入剖析基于并发AQS的重入锁(ReentrantLock)及其Condition实现原理	
* Android版本的"Wannacry"文件加密病毒样本分析(附带锁机)	
* 工作与生活真的可以平衡吗？	
* 《Real-Time Rendering 3rd》提炼总结——高级着色：BRDF及相关技术	
* 《三体》读后思考-泰勒展开/维度打击/黑暗森林	

最新评论	
理解Semaphore及其用法详解 panxianzhan: 谢谢，当看到《Unix环境高级编程》说的"Otherwise, if the value of th...	
理解Semaphore及其用法详解 夜空中最亮的星_追梦人SYC: 谢谢楼主分享精神	
linux网络编程基本流程 poopooptt: 明明是tcp/ip的编程，非要说什么七层协议。	
Linux驱动程序开发 - 设备控制接 jackylongchen: 再次学习 TKS	
cdev_init()函数 fengweihao158: 顶一下！	
解读set_gpio_ctrl(GPIO_MODE_ flyingdon: 2.6.32.2内核中对2410 GPIO进行操作的几个宏是定义在linux/include/asm-...	
解读set_gpio_ctrl(GPIO_MODE_ ww1236542: 我的内核代码中怎么没有#include/asm-arm/arch-s3c2410/s3c2410.h...	
一步步设计自己的驱动程序 (转flyingdon: 呵呵，上面的代码你可以自己综合一下，不过需要注意的是，内核版本不同的话需要再修改一下头函数，祝你好运...	
一步步设计自己的驱动程序 (转liuxiaoboxhn: 详细的代码 能给个吗？	
一步步设计自己的驱动程序 (转liuxiaoboxhn: 您能给个纤细完整的代码吗？	

所以，mutex就是一个binary semaphore（值就是0或者1）。但是他们的区别又在哪里呢？主要有两个方面：

* 初始状态不一样：mutex的初始值是1（表示锁available），而semaphore的初始值是0（表示unsigaled的状态）。随后的操作基本一样。mutex_lock和sem_post都把值从0变成1，mutex_unlock和sem_wait都把值从1变成0（如果值是零就等待）。初始值决定了：虽然mutex_lock和sem_wait都是执行V操作，但是sem_wait将立刻将当前线程block住，直到有其他线程 post；mutex_lock在初始状态下是可以进入的。

* 用法不一样（对称 vs. 非对称）：这里说的是“用法”。Semaphore实现了signal，但是mutex也有signal（当一个线程lock后另外一个线程 unlock，lock住的线程将收到这个signal继续运行）。在mutex的使用中，模型是对称的。unlock的线程也要先lock。而 semaphore则是非对称的模型，对于一个semaphore，只有一方post，另外一方只wait。就拿上面的厕所理论来说，mutex是一个钥匙不断重复的使用，传递在各个线程之间，而semaphore是一方不断的制造钥匙，而供另外一方使用（另外一方不用归还）。

前面的实验证明，mutex确实能够做到post和wait的功能，只是大家不用而已，因为它是“mutex”不是semaphore。

下面给出一个例子：

要 让一个thread在背景不断的执行，最简单的方式就是在该thread执行无穷回圈，如while(1) {}，这种写法虽可行，却会让CPU飙高到100%，因为CPU一直死死的等，其实比较好的方法是，背景平时在Sleep状态，当前景呼叫背景时，背景马上被唤醒，执行该做的事，做完马上Sleep，等待前景呼叫。当背景sem_wait()时，就是马上处于Sleep状态，当前景sem_post() 时，会马上换起背景执行，如此就可避免CPU 100%的情形了。

```
/**/
(C) OOMusou 2006 http://oomusou.cnblogs.com

Filename : pthread_create_semaphore.cpp
Compiler : gcc 4.10 on Fedora 5 / gcc 3.4 on Cygwin 1.5.21
Description : Demo how to create thread with semaphore in Linux.
Release : 12/03/2006
Compile : g++ -lpthread pthread_create_semaphore.cpp
*/

#include <stdio.h> // printf(),
#include <stdlib.h> // exit(), EXIT_SUCCESS
#include <pthread.h> // pthread_create(), pthread_join()
#include <semaphore.h> // sem_init()

sem_t binSem;

void* helloWorld(void* arg);

int main() {
    // Result for System call
    int res = 0;

    // Initialize semaphore
    res = sem_init(&binSem, 0, 0);
    if (res) {
        printf("Semaphore initialization failed!!/n");
        exit(EXIT_FAILURE);
    }

    // Create thread
    pthread_t thdHelloWorld;
    res = pthread_create(&thdHelloWorld, NULL, helloWorld, NULL);
    if (res) {
        printf("Thread creation failed!!/n");
        exit(EXIT_FAILURE);
    }

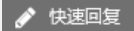
    while(1) {
```

```
// Post semaphore
sem_post(&binSem);
printf("In main, sleep several seconds.\n");
sleep(1);
}

// Wait for thread synchronization
void *threadResult;
res = pthread_join(thdHelloWorld, &threadResult);
if (res) {
    printf("Thread join failed!!\n");
    exit(EXIT_FAILURE);
}

exit(EXIT_SUCCESS);
}

void* helloWorld(void* arg) {
    while(1) {
        // Wait semaphore
        sem_wait(&binSem);
        printf("Hello World\n");
    }
}
```



编译运行：

```
[root@localhost semaphore]# gcc semaphore.c -lpthread
[root@localhost semaphore]# ./a.out
In main, sleep several seconds.
Hello World
In main, sleep several seconds.
Hello World
In main, sleep several seconds.
Hello World
In main, sleep several seconds.
Hello World
```

semaphore

信号量(Semaphore)，有时被称为信号灯，是在多线程环境下使用的一种设施，它负责协调各个线程，以保证它们能够正确、合理的使用公共资源。

什么是信号量(Semaphore0)

Semaphore分为单值和多值两种，前者只能被一个线程获得，后者可以被若干个线程获得。

以一个停车场是运作为例。为了简单起见，假设停车场只有三个车位，一开始三个车位都是空的。这是如果同时来了五辆车，看门人允许其中三辆不受阻碍的进入，然后放下车栏，剩下的车则必须在入口等待，此后来的车也都不得不在入口处等待。这时，有一辆车离开停车场，看门人得知后，打开车栏，放入一辆，如果又离开两辆，则又可以放入两辆，如此往复。

在这个停车场系统中，车位是公共资源，每辆车好比一个线程，看门人起的就是信号量的作用。

更进一步，信号量的特性如下：信号量是一个非负整数（车位数），所有通过它的线程（车辆）都会将该整数减一（通过它当然是为了使用资源），当该整数值为零时，所有试图通过它的线程都将处于等待状态。在信号量上我们定义两种操作：Wait（等待）和Release（释放）。当一个线程调用Wait等待操作时，它要么通过然后将信号量减一，要么一自等下去，直到信号量大于一或超时。Release（释放）实际上是在信号量上执行加操作，对应于车辆离开停车场，该操作之所以叫做“释放”是应为加操作实际上是释放了由信号量守护的资源。

实现

大家都知道，.Net Framework类库中提供的线程同步设施包括：

Monitor，AutoResetEvent，ManualResetEvent，Mutex，ReadWriteLock和InterLock。其中AutoResetEvent，ManualResetEvent，Mutex派生自WaitHandler，它们实际上是封装了操作系统提供的内核对象。而其它的应当是在.Net虚拟机中土生土长的。显然来自操作系统内核对象的设施使用起来效率要差一些。不过效率并不是我们这里要考虑的问题，我们将使用两个Monitor和一个ManualResetEvent对象来模拟一个信号量。


关闭

代码如下：

```
public class Semaphore
{
    private ManualResetEvent waitEvent = new ManualResetEvent(false);
    private object syncObjWait = new object();
    private int maxCount = 1; file://最大资源数
    private int currentCount = 0; file://当前资源数
    public Semaphore()
    {
    }
    public Semaphore( int maxCount )
    {
        this.maxCount = maxCount;
    }
    public bool Wait()
    {
        lock( syncObjWait ) file://只能一个线程进入下面代码
        {
            bool waitResult = this.waitEvent.WaitOne(); file://在此等待资源数大于零
            if( waitResult )
            {
                lock( this )
                {
                    if( currentCount > 0 )
                    {
                        currentCount--;
                        if( currentCount == 0 )
                        {
                            this.waitEvent.Reset();
                        }
                    }
                }
            }
            else
            {
                System.Diagnostics.Debug.Assert( false, "Semaphore is not allow current count < 0" );
            }
        }
        return waitResult;
    }
}

/**/// <summary>
/// 允许超时返回的 Wait 操作
/// </summary>
/// <param name="millisecondsTimeout"></param>
/// <returns></returns>
public bool Wait( int millisecondsTimeout )
{
    lock( syncObjWait ) // Monitor 确保该范围类代码在临界区内
    {
        bool waitResult = this.waitEvent.WaitOne(millisecondsTimeout,false);
        if( waitResult )
        {
            lock( this )
            {
```

 快速回复

 我要收藏

```
if( currentCount > 0 )
{
    currentCount--;
    if( currentCount == 0 )
    {
        this.waitEvent.Reset();
    }
}
else
{
    System.Diagnostics.Debug.Assert( false, "Semaphore is not allow current count to be 0." );
}
}
return waitResult;
}
}
public bool Release()
{
    lock( this ) // Monitor 确保该范围类代码在临界区内
    {
        currentCount++;
        if( currentCount > this.maxCount )
        {
            currentCount = this.maxCount;
            return false;
        }
        this.waitEvent.Set(); file://允许调用Wait的线程进入
    }
    return true;
}
}
```

快速回复

☆ 我要收藏



顶3

踩1

- 上一篇 Kmalloc
- 下一篇 fseek()/ftell()/rewind()/lseek()/fstat()

相关文章推荐

- SQLite之大数据量批量入库
- Java之Semaphore
- Java之Final
- Java并发之Semaphore详解
- Java并发编程中Semaphore的用法

- 信号量Semaphore初探
- 深入理解Semaphore
- Semaphore 的理解
- Semaphore的介绍和使用
- 内核同步机制-信号量 (semaphore)

猜你在找

- 【直播】机器学习&深度学习系统实战（唐宇迪）
- 【直播】Kaggle 神器：XGBoost 从基础到实战（冒教授）
- 【直播】计算机视觉原理及实战（屈教授）
- 【直播】机器学习之矩阵（黄博士）
- 【直播】机器学习之数学基础
- 【直播】深度学习30天系统实训（唐宇迪）
- 【直播回放】深度学习基础与TensorFlow实践（王琛）
- 【直播】机器学习之凸优化（马博士）
- 【直播】机器学习之概率与统计推断（冒教授）
- 【直播】TensorFlow实战进阶（智亮）

查看评论

2楼 [panxianzhan](#) 2016-03-24 18:19发表



谢谢，当看到《Unix环境高级编程》说的"Otherwise, if the value of the semaphore is 0, the process goes to sleep until... 句之后我很敏感地觉得信号量难道是只能用于进程间通信吗？如果是同一个process的不同线程去使用，那sleep那不是悲剧了？

看到你的例子，暂时打消了这个疑惑，我会做个实验尝试一下的

快速回复

我要收藏

1楼 [夜空中最亮的星_追梦人SYC](#) 2016-01-04 16:28发表



谢谢楼主分享精神

您还没有登录,请[登录](#)或[注册](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved