**实验四：信号量的实现**

1. **实验目的**

* 加深对进程同步与互斥概念的认识；
* 掌握信号量的实现原理；
* 在实际操作系统中实践信号量的实现，并加以实际应用。

1. **实验内容：**

* 实现信号量
* 使用信号量，解决生产者消费者问题

1. **实验准备**

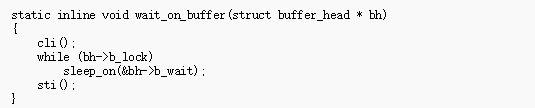
**①关于Linux 0.11的信号量**

Linux0.11 内核中并没有定义信号量，由Dijkstra的信号量设计可以看出，信号量的实质就是一个锁，它和一般的锁（一般的锁在现代Linux中称为自旋锁）区别在于信号量可以让进程睡眠在信号量上来让出CPU，所以信号量不是自旋锁。

Linux0.11中虽然没有信号量，但是Linux 0.11 也是一个支持并发和虚拟内存的现代操作系统。它提供了一定的锁机制来实现多个进程访问共享资源的互斥和同步。

以读取磁盘缓存为例，因为磁盘缓存是一段内存缓存。进程提出的磁盘访问请求首先要到磁盘缓存中去找，如果找到直接返回；如果没有找到则申请一段空闲的磁盘缓存，以这段磁盘缓存为参数发起磁盘读写请求，发起请求后进程要睡眠等待（因为磁盘读写很慢，应该让出CPU让其他进程执行）。这种方法是许多操作系统（包括现代Linux、UNIX等）采用的办法。对于这种方法：多个进程共同操作磁盘缓存集合，进程会在操作过程中让出CPU。这两个特性使得在操作磁盘缓存时需要考虑互斥问题，所以其中必定用到了锁。

如：



这是linux中fs/buffer.c中的一个函数，cli(),sli()用来打开关闭中断，linux以这种形式实现只有一个进程进入。

并且我们可以看出sleep\_on和wake\_up的操作都是现在一个struct task\_struck\*上，即睡眠或者唤醒在进程的PCB块上，我们可以用这个方法来实现信号量。

**②信号量复习**

一个信号量：



信号量的运作方式：

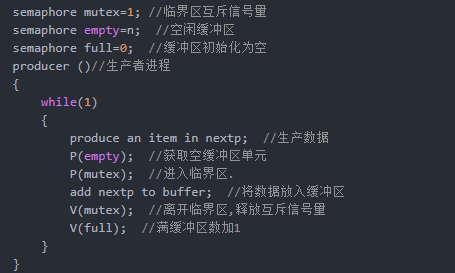
* 初始化，给与它一个非负数的整数值。
* 运行 P（wait()），信号量S的值将被减少。企图进入临界区块的进程，需要先运行 P（wait()）。当信号量S减为负值时，进程会被挡住，不能继续；当信号量S不为负值时，进程可以获准进入临界区块。
* 运行 V（又称signal()），信号量S的值会被增加。结束离开临界区块的进程，将会运行 V（又称signal()）。当信号量S不为负值时，先前被挡住的其他进程，将可获准进入临界区块。

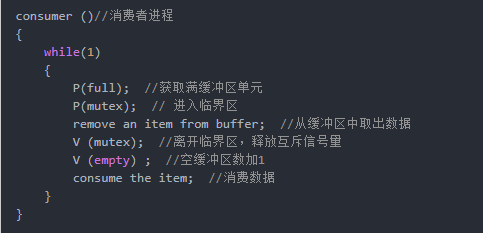
**③关于生产者消费者问题：**

生产者消费者问题（英语：Producer-consumer problem），也称有限缓冲问题（英语：Bounded-buffer problem），是一个多线程同步问题的经典案例。该问题描述了共享固定大小缓冲区的两个线程——即所谓的“生产者”和“消费者”——在实际运行时会发生的问题。生产者的主要作用是生成一定量的数据放到缓冲区中，然后重复此过程。与此同时，消费者也在缓冲区消耗这些数据。该问题的关键就是要保证生产者不会在缓冲区满时加入数据，消费者也不会在缓冲区中空时消耗数据。

要解决该问题，就必须让生产者在缓冲区满时休眠（要么干脆就放弃数据），等到下次消费者消耗缓冲区中的数据的时候，生产者才能被唤醒，开始往缓冲区添加数据。同样，也可以让消费者在缓冲区空时进入休眠，等到生产者往缓冲区添加数据之后，再唤醒消费者。通常采用进程间通信的方法解决该问题。如果解决方法不够完善，则容易出现死锁的情况。出现死锁时，两个线程都会陷入休眠，等待对方唤醒自己。该问题也能被推广到多个生产者和消费者的情形。

**一种解决方法的伪代码：**



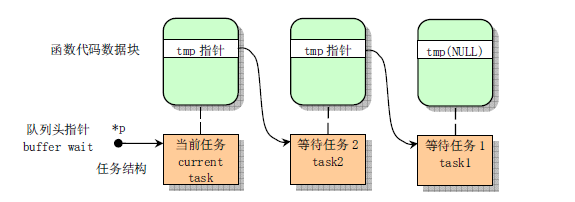
****

**④关于sleep\_on的理解:**

在上一个实验中，我们在sched.c文件中曾经见到过sleep\_on函数，当时只清楚其答题的功能但是没有理解其精华。

sleep\_on和wake\_up这两个函数是两个理解难度的函数。首先sleepon的传参是一个进程结构体的双重指针，这个双重指针一般是针对一个资源的等待队列。这个队列是如何实现的呢。

他使用了tmp来构成一个隐式等待队列：



比如有A、B、C三个进程争夺一个资源。

A进程sleep\_on，p指向A，tmp指向NULL

B进程sleep\_on，p指针指向B，tmp指向A

C进程sleep\_on,p指针指向C，tmp指向B

在进程改为可中断睡眠后，进入schedule函数重新调度，注意只有执行sleepon的进程在wakeup并且得到调度后才会继续执行schedule函数下的那句话，这句话用来唤醒tmp指针指向的任务，然后链式唤醒。这里要非常注意的是要以多进程的角度看待这个过程的执行。

1. **实验步骤**

**①实现信号量：**

由于信息量的使用是系统调用，所以我们要仿照实验2的步骤添加几个系统调用函数。

1. **思考**