Lab4 进程同步

俞星凯 171830635

2651904866@qq.com

**实验目的**

1. 实现一个简单的生产者消费者程序。
2. 介绍基于信号量的进程同步机制。

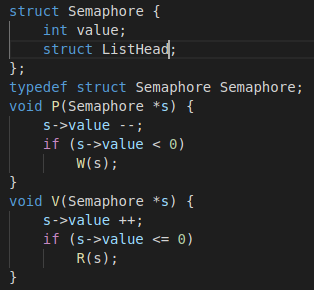
**实验内容**

1. 内核：提供基于信号量的进程同步机制，并提供系统调用sem\_init、sem\_post、sem\_wait、sem\_destroy。
2. 库：对上述系统调用进行封装。
3. 用户：对上述库函数进行测试。

**背景知识**

1. 信号量机制

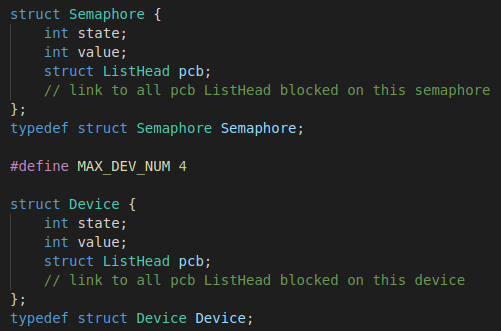
内核维护Semaphore这一数据结构，并提供P，V这一对原子操作，其中W用于阻塞进程自身在该信号量上，R用于释放一个阻塞在该信号量上的进程，其伪代码如下

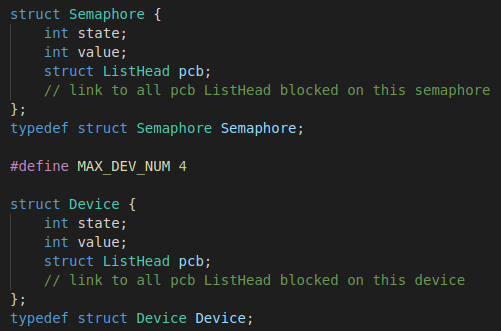


**实验过程**

1. 启动阶段

大体与Lab3类似，不同之处在于加入两个数组sem和dev用来实现进程同步，两个数组的元素类型是类似的，其中，state表示该元素是否有效，value表示信号量的值，为正时表示在封锁前对s可施行的P操作，为负时其绝对值表示在s队列中等待的进程个数，pcb则是一个双向链表，用于记录执行P操作时被阻塞的进程。

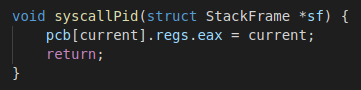




1. 系统调用
2. GETPID系统调用

为了方便区分当前正在运行的进程，实验4要求先实现一个getpid系统调用用于返回当前进程标识ProcessTable.pid，不允许调用失败。

getpid的实现较为简单，添加宏定义#define SYS\_PID 7，并在irqhandle.c添加一个系统调用处理函数，利用eax返回pid即可。



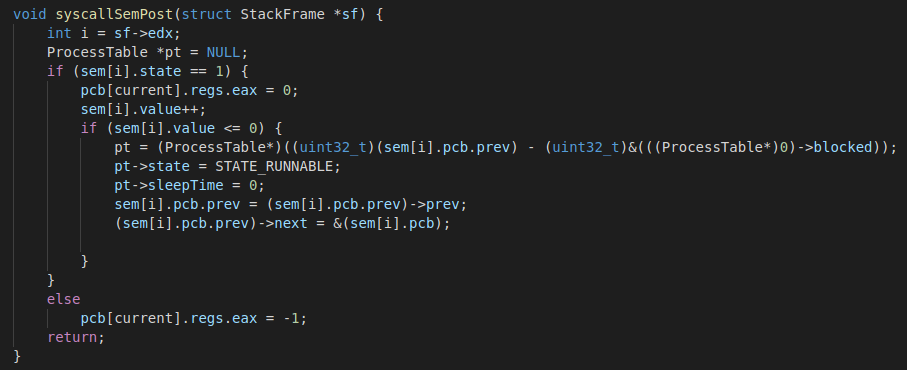
1. SEM\_INIT系统调用

sem\_init系统调用用于初始化信号量，其中参数value用于指定信号量的初始值，初始化成功则返回0，指针sem指向初始化成功的信号量，否则返回-1。

sem\_init的实现流程是在数组sem中查找一个未使用的信号量，根据参数初始化信号量的值value，并设置好双向链表pcb。在框架代码中已经给出。

1. SEM\_POST系统调用

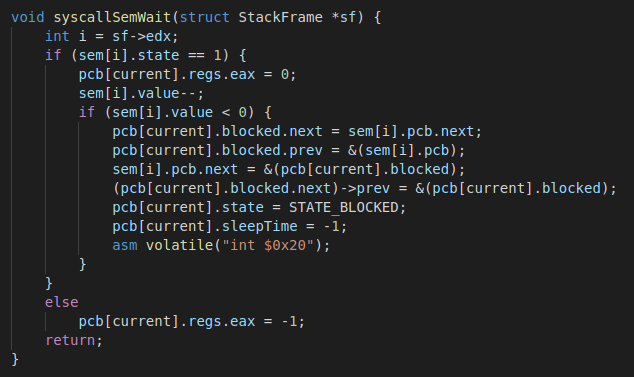
sem\_post系统调用对应信号量的V操作，其使得sem指向的信号量的value增一，若value取值不大于0，则释放一个阻塞在该信号量上进程（即将该进程设置为就绪态），若操作成功则返回0，否则返回-1。



sem\_post首先检查参数对应的信号量state是否为1，若为0则报错，否则将value加一，若value<=0，说明有进程被阻塞，应该释放。因为pcb的prev是最先被阻塞的进程，所以将它从双向链表取出。

1. SEM\_WAIT系统调用

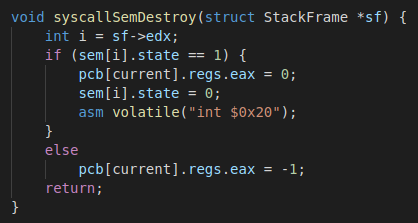
sem\_wait系统调用对应信号量的P操作，其使得sem指向的信号量的value减一，若value取值小于0，则阻塞自身，否则进程继续执行，若操作成功则返回0，否则返回-1。



sem\_wait同样检查state，然后将value减一，若value<0，说明进程应该被阻塞，将其加入双向链表pcb的next，并设置pcb[current]的state和sllepTime，最后激发时钟中断重新调度。

1. SEM\_DESTROY系统调用

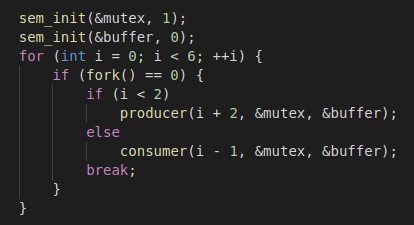
sem\_destroy系统调用用于销毁sem指向的信号量，销毁成功则返回0，否则返回-1，若尚有进程阻塞在该信号量上，可带来未知错误。



sem\_destroy直接将信号量的state清零，并触发时钟中断重新调度即可。

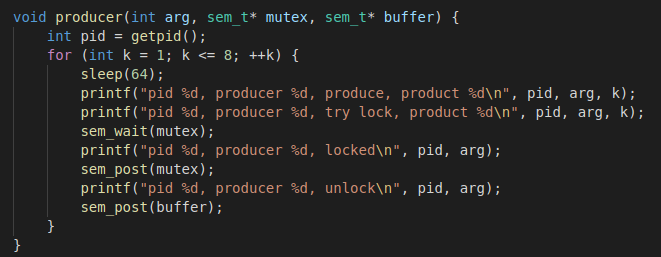
1. 信号量解决生产者消费者问题
2. main函数

main函数需要准备两个信号量mutex(用于互斥)和buffer(用于生产者消费者同步)，value分别为1和0，然后循环fork出6个子进程，根据fork的返回值，若为0说明是子进程，调用相应的生产者消费者函数，并跳出循环，否则是父进程，继续循环。



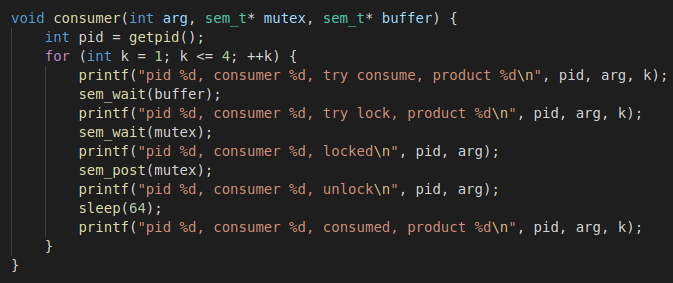
1. 生产者进程

生产者循环生产，生产过程用sleep(64)模拟，生产完成打印produce信息，接着尝试获取mutex实现互斥访问，最后对buffer执行V操作表示buffer内已经有产品了。



1. 消费者进程

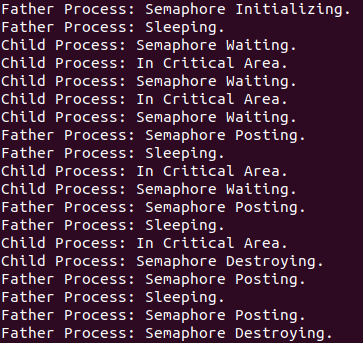
消费者循环消费，打印一条try consume信息，并对buffer执行P操作，此时若buffer为空则被阻塞，接着获取mutex访问临界区，最后用sleep(64)模拟消费过程，在完成消费后打印consumed信息。



**实验结果**

1. 用户程序测试

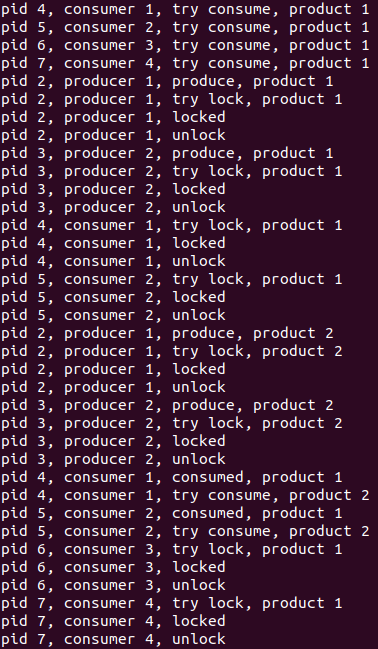
由框架代码给出的对系统调用和库函数的测试部分的实验结果如下



对执行过程的分析如下

1. 父进程初始化信号量，fork出子进程，打印sleeping后便去睡眠。
2. 由于信号量的值为2，子进程可以进入两次关键区，在第三次时被阻塞。
3. 父进程苏醒，释放信号量，又去睡眠
4. 被阻塞的子进程释放，再次进入关键区，又被阻塞。
5. 同4。
6. 被阻塞的子进程释放，销毁信号量，退出。
7. 只有父进程运行，最终销毁信号量。
8. 生产者消费者

生产者消费者进程的执行结果较长，截取部分



对执行结果分析如下

1. 生产者1、2均执行sleep模拟生产过程，消费者1、2、3、4执行try consume但由于buffer为空被阻塞。
2. 生产者1、2各完成生产一个产品。
3. 消费者1、2各开始消费一个产品。
4. 生产者1、2各完成生产一个产品。
5. 消费者1、2各完成消费第一个产品，尝试消费第二个产品被阻塞。
6. 消费者3、4各开始消费一个产品。

**实验收获**

1. 了解了基于信号量的进程同步机制。
2. 提高了编写进程同步代码的能力。