哈工大操作系统-L18信号量的代码实现

哈工大操作系统-L18信号量的代码实现
1.信号量的简单实现
2.Linux.11的实现

1.信号量的简单实现

```
Producer(item) {
    P(empty);
        ...
    V(full); }
```
生产者消费者的模型

用户态程序

producer.c

```
main(){
(1)sd=sem_open("empty");
(2)for(i=1 to 5)
    sem_wait(sd);
    write(fd,&i,4);
```
想在磁盘中写出5个数

```
sem.c //进入内核
typedef struct {
  char name[20];int value;
  task_struct * queue;
} semtable[20];
```
信号量的名字 信号量的值 信号量的队列 构建这样的信号量的结构体

```
sys_sem_wait(int sd){
  cli();
  if(semtable[sd].value
-- < 0){设置自己为阻塞;将自
己加入semtable[sd].queue
中;schedule();}
  sti(); }
```
中断保证操作的原子性 中断保证操作的原子性

如果没有可用资源，要等待，自己进入等待队列

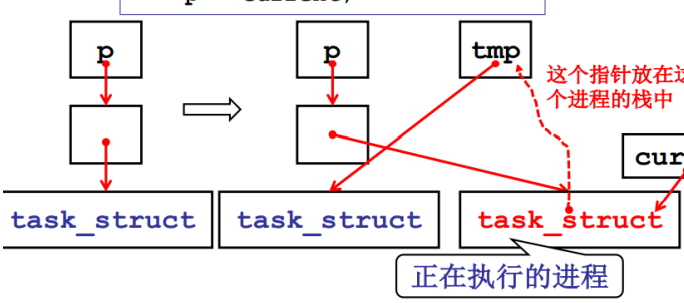
```
sys_sem_open(char *name)
{
  在semtable中寻找name对上的;
  没找到则创建;
  返回对应的下标; }
```

2.Linux.11的实现

Linux 0.11 sleep_on形成的队列

```
sleep_on(struct task_struct **p)
p是一个指向task_struct结构体的指针的指针
    struct task_struct *tmp;
    tmp = *p;
    *p = current;
```

局部变量保存在栈中
因为是内核态程序
这里的局部变量保存在内核栈中
因此，这个tmp被保存在当前进程的内核栈中
而后，又把tmp设置成沉睡队列的队首，即当前进程的tmp指向旧的沉睡队列的队首
而当前的沉睡队列的队首指针，变成了当前进程
因此，这样就构成了一个队列
从沉睡队列可以找到第一个沉睡进程，在第一个沉睡进程的PCB中
要找找到tmp而tmp指向的就是下一个沉睡进程



在多个进程等待同一资源时，隐式构建一个等待队列。

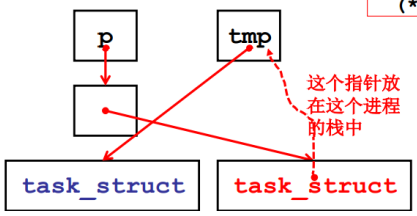
如何从Linux 0.11的这个队列中唤醒?

```
static void read_intr(void){
  ...
  end_request(1);
```
磁盘中断

```
end_request(int uptodate){
  ... 磁盘中断结束时，会唤醒进程
  unlock_buffer(CURRENT->bh);
```

```
unlock_buffer(struct
buffer_head * bh){
    bh->b_lock=0;
    wake_up(&bh->b_wait);}
```
唤醒函数解锁(将信号量变为0)

```
wake_up(struct task_struct
**p){ if (p && *p) {
  (**p).state=0; *p=NULL;}}
```



问题：这个队列是怎么唤醒的?

问题：while(lock)?

```
schedule();
if (tmp)
  tmp->state=0;
```
因为沉睡之前到了这里。因此醒来的时候也是到这里
此时进程会将下一个进程也唤醒

这是sleep_on的最后三句

以此类推，把所有的进程都唤醒

? 没有很懂

然后唤醒沉睡队列的第一个进程(沉睡队列的第一个函数变为就绪态)

从Linux 0.11那里学点东西...

读磁盘块

```
bread(int dev,int block){
   struct buffer_head * bh;
   ll_rw_block(READ,bh);
   wait_on_buffer(bh);
```

- 启动磁盘读以后睡眠，等待磁盘读完由磁盘中断将其唤醒，也是一种同步

```
lock_buffer(buffer_head*bh)
{cli();
while(bh->b_lock)???
    sleep_on(&bh->b_wait);
bh->b_lock = 1;
sti(); }
```

```
void sleep_on(struct task_struct **p){
   struct task_struct *tmp;
   tmp = *p;
   *p = current;
   current->state = TASK_UNINTERRUPTIBLE;
    schedule();
    if (tmp)
      tmp->state=0;}
```

问题：这个世界上最隐蔽的队列张什么样子?

使用while()是因为，wakeup把所有进程都唤醒，然后会调用schedule进行调度，因此还要while一下看看是不是该我了

这种while的方式与前面的if但信号量有负数的实现方式有不同。