操作系统 Operating Systems

L18 信号量的代码实现

Coding Semaphore

授课教师：李治军
lizhijun_os@hit.edu.cn
综合楼411室

可以操刀了—从纸上到实际

```
Producer(item) {
    P(empty);
        ...
    V(full); }
```

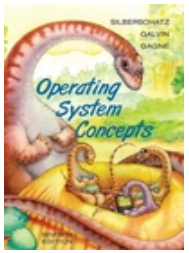
用户态程序

producer.c

```
main(){
(1)sd=sem_open("empty");
(2)for(i=1 to 5)
     sem_wait(sd);
     write(fd,&i,4);
```

```
sem.c //进入内核
typedef struct {
  char name[20];int value;
   task_struct * queue;
} semtable[20];
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; }
```

```
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(); }
```



从Linux 0.11那里学点东西...

读磁盘块

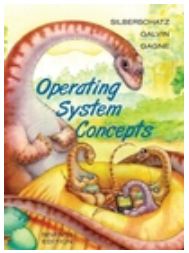
```
bread(int dev,int block){
   struct buffer_head * bh;
   ll_rw_block(READ,bh);
   wait_on_buffer(bh);
```

- 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?

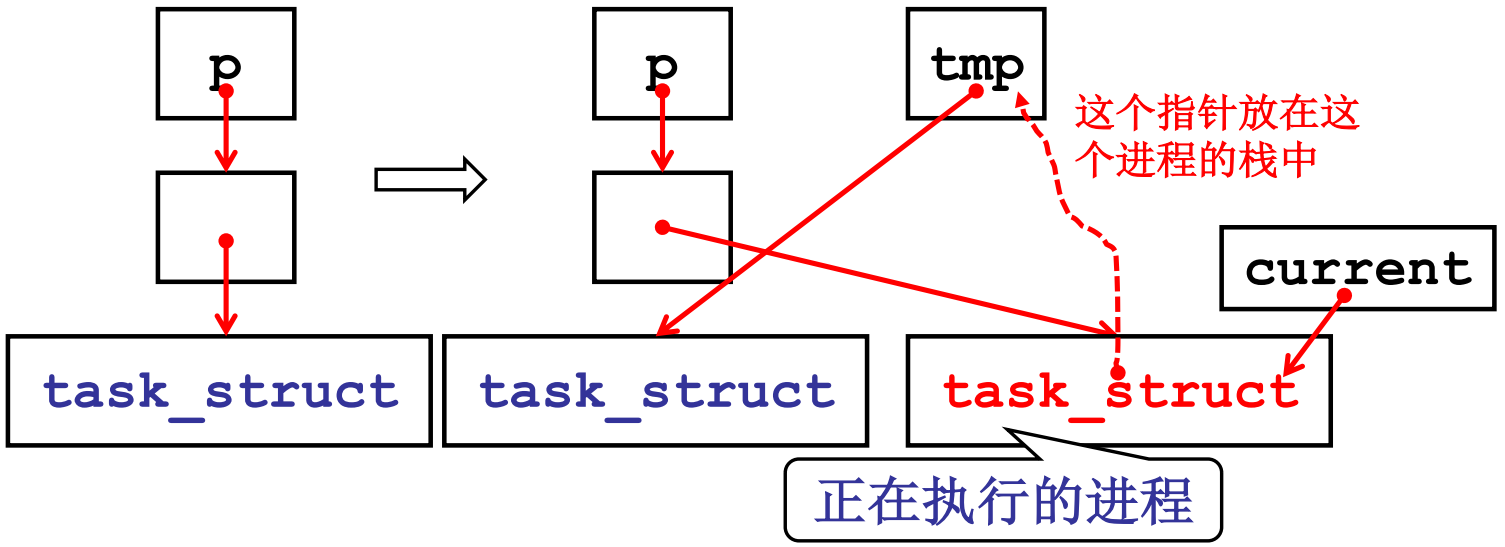


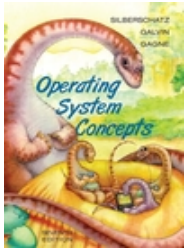
Linux 0.11 sleep_on形成的队列

```
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;
```





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

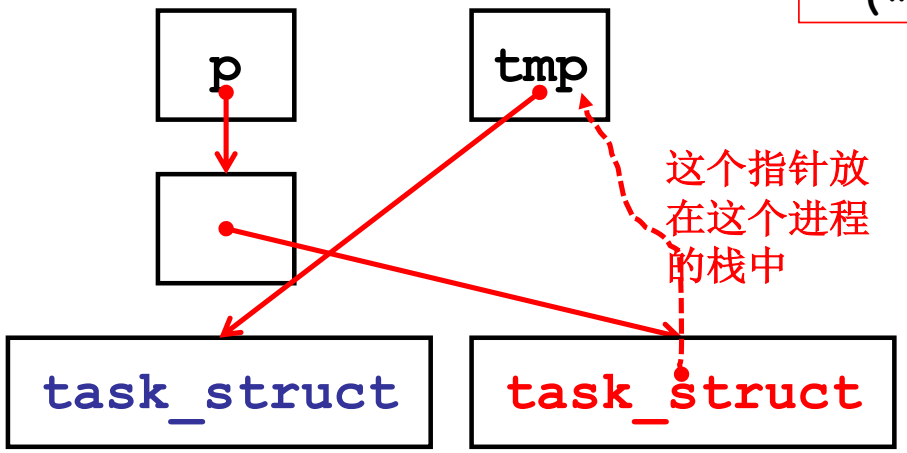
```
end_request(int uptodate){
  ...
  unlock_buffer(CURRENT->bh);
```

```
unlock_buffer(struct
buffer_head * bh){
    bh->b_lock=0;
    wake_up(&bh->b_wait);}
```

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

```
schedule();
if (tmp)
  tmp->state=0;
```

这是sleep_on的最后三句



问题：这个队列是怎么唤醒的?

问题：while(lock)?