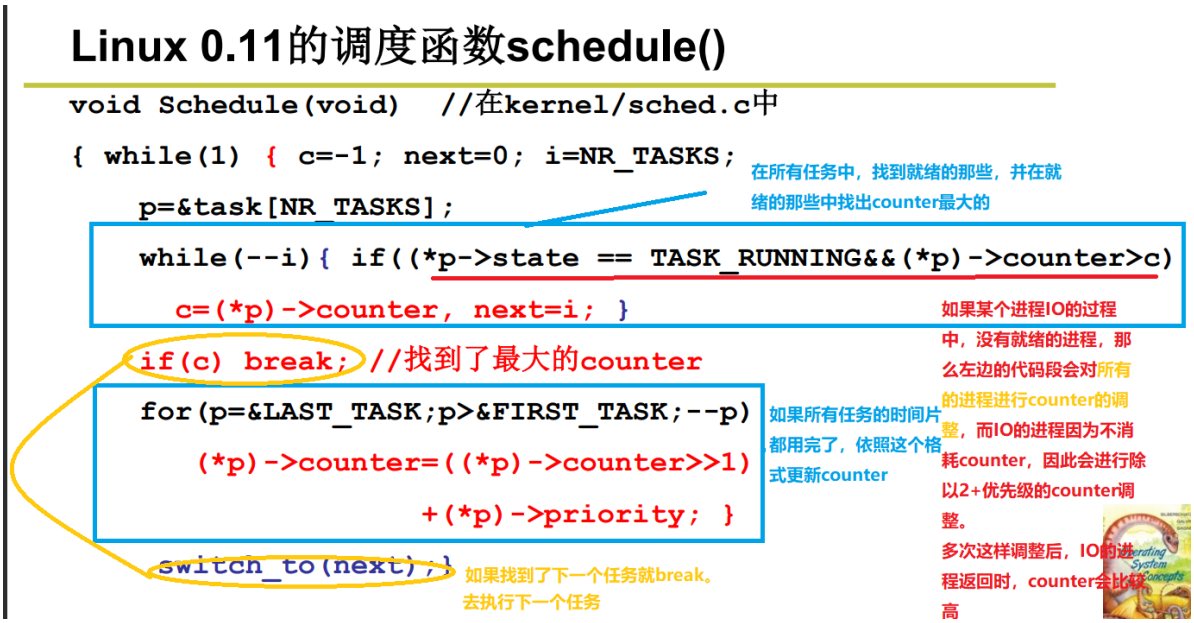
哈工大操作系统-L15一个实际的schedule函数

讲述Linux 0.11的调度函数schedule()

counter时间片+SFJ调度(几何级数更新counter保证最长为2P)



时间片的作用

counter的作用: 时间片

```
void do_timer(...)    //在kernel/sched.c中
{   if((--current->counter>0) return;
    current->counter=0;
    schedule(); }
```

```
_timer_interrupt: //在kernel/system_call.s中
   ...
   call _do_timer
```

每到一个时间点，调用一次_do_timer，对counter减少一个时间单位，起到时间片的作用。如果counter用完了，那么调用别的应用。

```
void sched_init(void) {
   set_intr_gate(0x20, &timer_interrupt);
```

- counter是典型的时间片，所以是轮转调度，保证了响应

优先级的作用

优先级使得IO的程序优先调度

counter的另一个作用: 优先级

```
while(--i){ if((*p->state == TASK_RUNNING&&(*p)->counter>c)
        c=(*p)->counter, next=i; }
```

- 找counter最大的任务调度，counter表示了优先级

```
for(p=&LAST_TASK;p>&FIRST_TASK;--p)
   (*p)->counter=((*p)->counter>>1)+(*p)->priority; }
```

- counter代表的优先级可以动态调整

阻塞的进程再就绪以后优先级高于非阻塞进程，为什么？

进程为什么会阻塞？I/O，正是前台进程的特征

而阻塞队列中的时间片尚未用完，此时也参与时间片的分配，除以2加初始值的分配方式会使得阻塞队列中的进程时间片慢慢增多

几何级数保证了时间片最长为2P

- 一直按照时间片轮转，且时间片长度有界这样能保证后台程序不会饥饿
- 不断的短时间片轮转，也使得短作业优先完事，这样也近似了SJF
- IO的优先级高也照顾了前台进程
- 只使用了一个counter就能维护

counter作用的整理

可能有个疑问，如果IO永远阻塞下去，那么阻塞的进程一旦不阻塞了，时间片不就会很长?
但是我们根据下面的c(t)和c(0)可知时间片是个几何级数，收敛到2P。因此时间片最长为2P

- counter保证了响应时间的界

$$c(t) = c(t-1)/2 + p$$
$$c(0) = p$$

$c(\infty) = ?$

- 经过IO以后，counter就会变大；IO时间越长，counter越大(为什么?)，照顾了IO进程，变相的照顾了前台进程
- 后台进程一直按照counter轮转，近似了SJF调度
- 每个进程只用维护一个counter变量，简单、高效

- **CPU调度: 一个简单的算法折中了大多数任务的需求，这就是实际工作的schedule函数**