

# 操作系统

# Operating Systems

## L15 一个实际的schedule函数

**schedule( )函数**

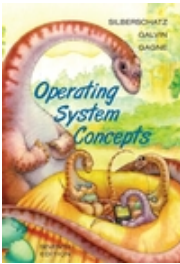
[lizhijun\\_os@hit.edu.cn](mailto:lizhijun_os@hit.edu.cn)

授课教师：李治军

综合楼411室

# Linux 0.11的调度函数schedule()

```
void Schedule(void) //在kernel/sched.c中
{ while(1) { c=-1; next=0; i=NR_TASKS;
    p=&task[NR_TASKS];
    while(--i){ if ((*p->state == TASK_RUNNING&&(*p)->counter>c)
        c=(*p)->counter, next=i; }
    if(c) break; //找到了最大的counter
    for (p=&LAST_TASK;p>&FIRST_TASK;--p)
        (*p)->counter= ((*p)->counter>>1)
            + (*p)->priority; }
    switch_to(next); }
```



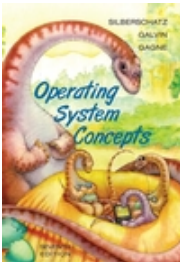
# counter的作用: 时间片

```
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
```

```
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



## 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; }
```

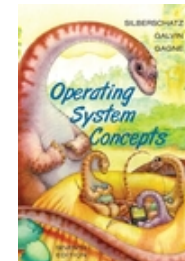
- 找counter最大的任务调度, counter表示了优先级

```
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



# counter作用的整理

- **counter**保证了响应时间的界

$$\begin{aligned}c(t) &= c(t-1)/2 + p \\ c(0) &= p\end{aligned}$$

$$c(\infty) = ?$$

- 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

