



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



L3-2. 多线程

宋卓然

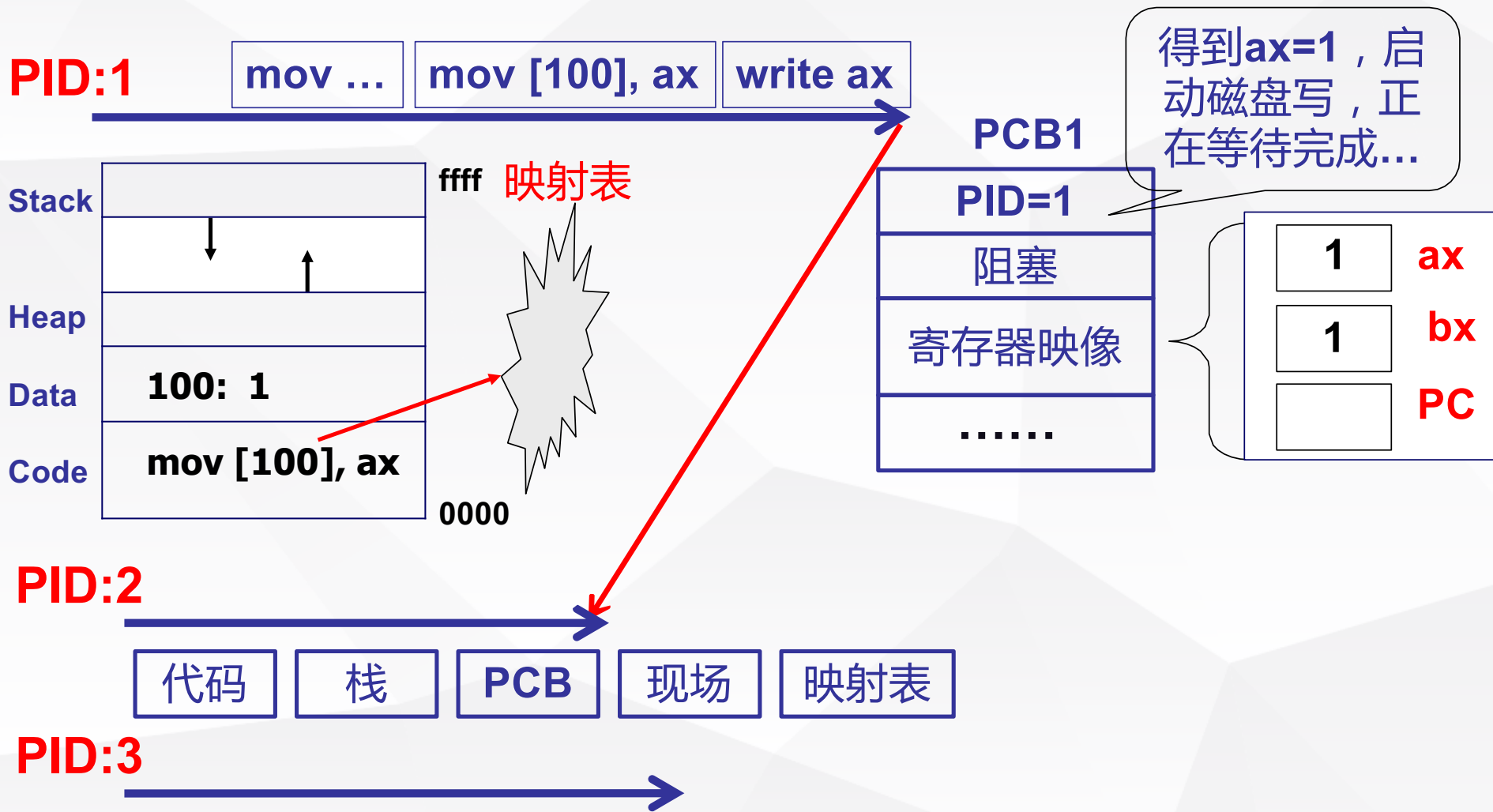
上海交通大学计算机系

songzhuoran@sjtu.edu.cn

饮水思源 · 爱国荣校



多进程是操作系统的基本图像

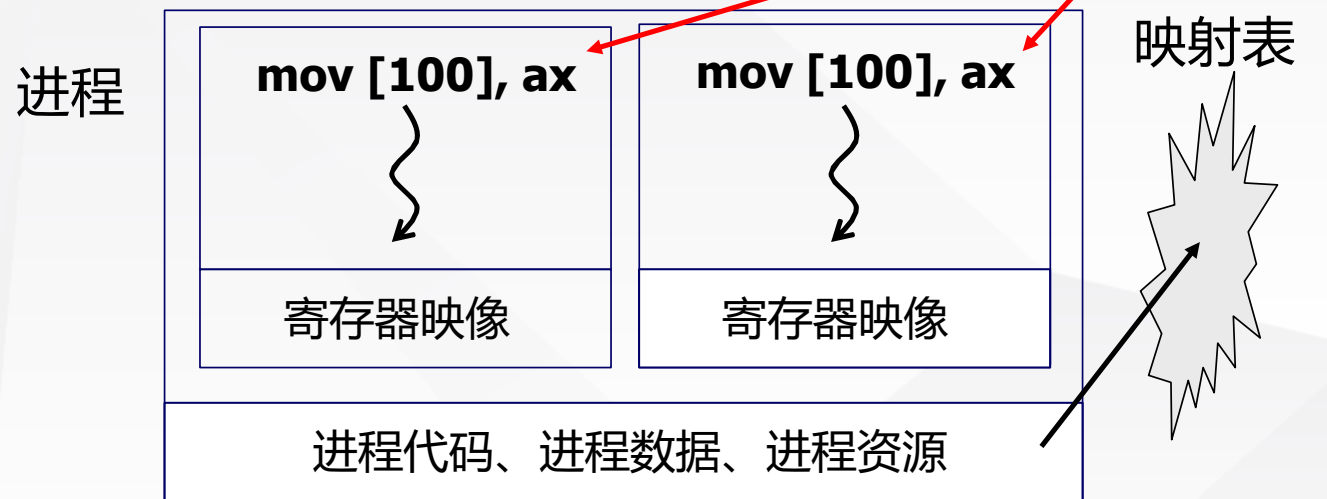




是否可以资源不动而切换指令序列？

■ 进程 = 资源 + 指令执行序列

- 将资源和指令执行分开
- 一个资源 + 多个指令执行序列



- 线程: 保留了并发的优点, 避免了进程切换代价
- 实质就是映射表不变而**PC**指针变



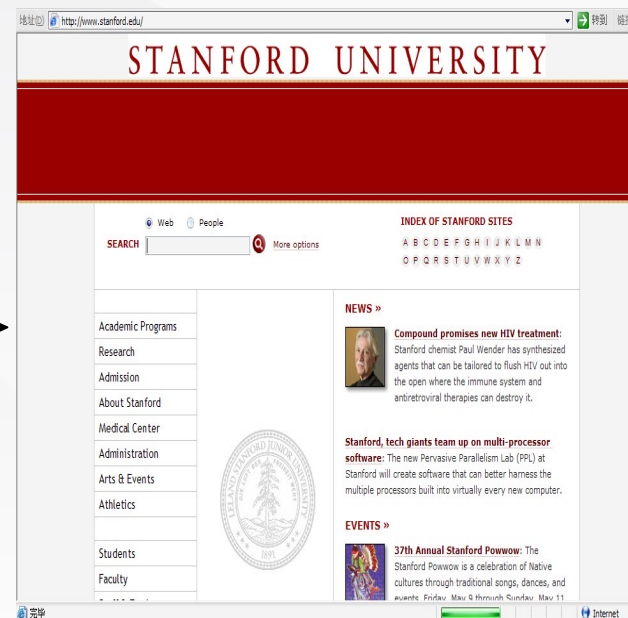
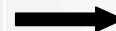
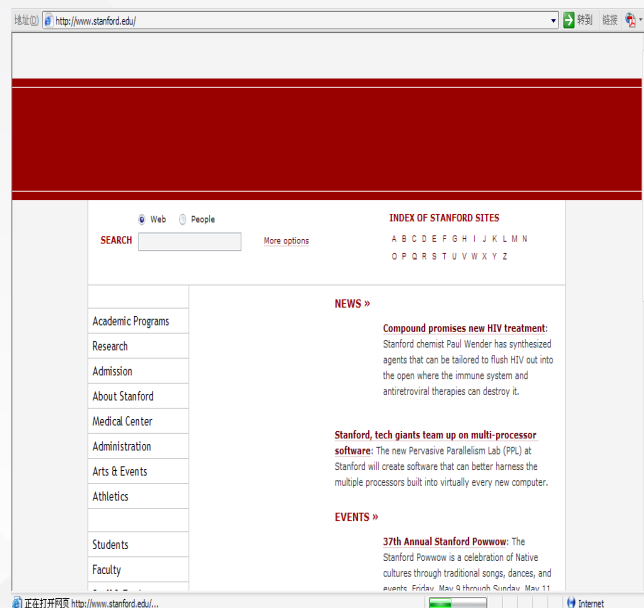
多个执行序列+一个地址空间是否实用？

一个网页浏览器

- 一个线程用来从服务器接收数据
- 一个线程用来显示文本
- 一个线程用来处理图片(如解压缩)
- 一个线程用来显示图片

这些线程要共享资源吗？

- 接收数据放在**100**处，显示时要读..
- 所有的文本、图片都显示在一个屏幕上





开始实现这个浏览器...

```
void WebExplorer()
{  char URL[] = "http://cms.hit.edu.cn";
   char buffer[1000];
   pthread_create(..., GetData, URL, buffer);
   pthread_create(..., Show, buffer); }

void GetData(char *URL, char *p){...};
void Show(char *p){...};
```





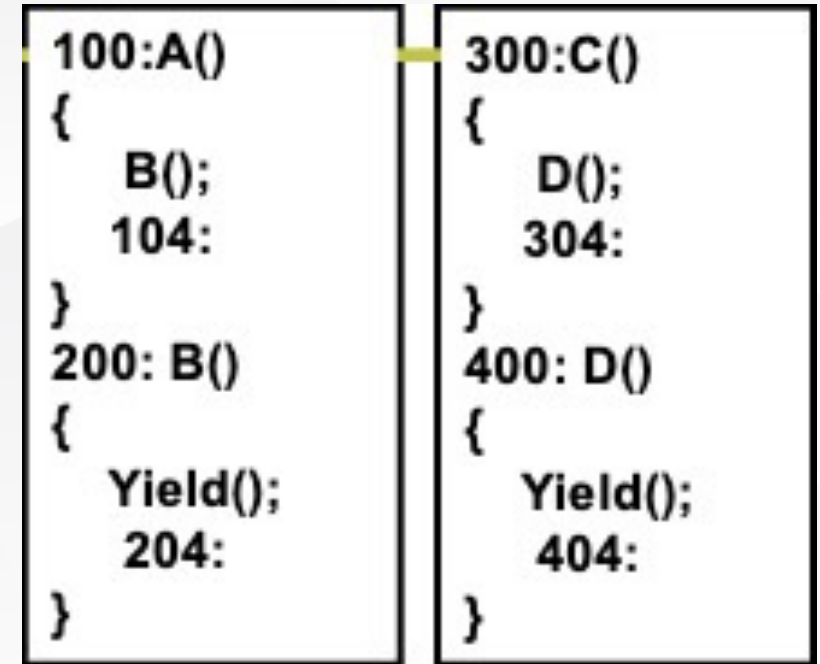
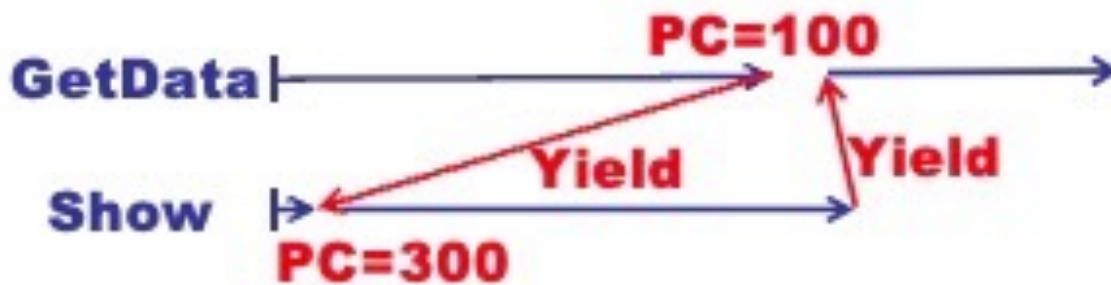
Create? Yield?

■核心是Yield...

- 切换线程
- **Create**是创造线程

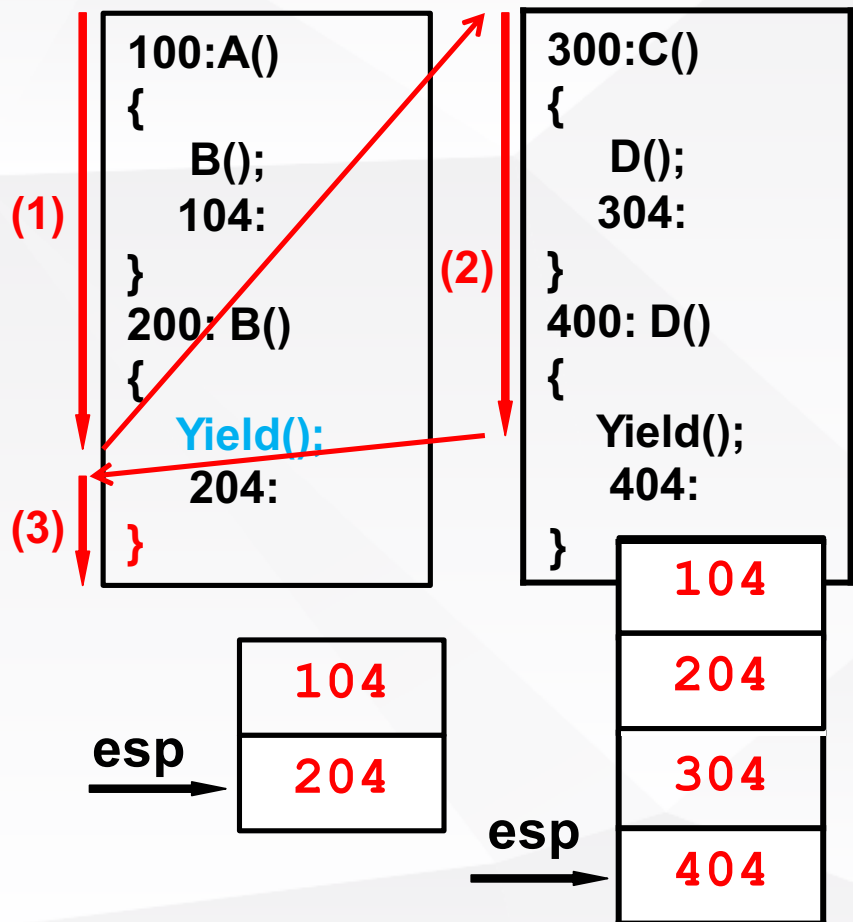
■仔细看Yield，就是100跳到300

样子弄明白了，剩下的就是写程序实现这个样子了...





两个执行序列与一个栈...



```
void Yield()  
{  
    找到300;  
    jmp 300;  
}
```

```
void Yield()  
{  
    找到 ?; 204  
    jmp ?;  
}
```

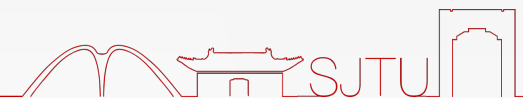
■ (3)再往下执行会怎么样?

花括号弹栈

■ 问题怎么解决?

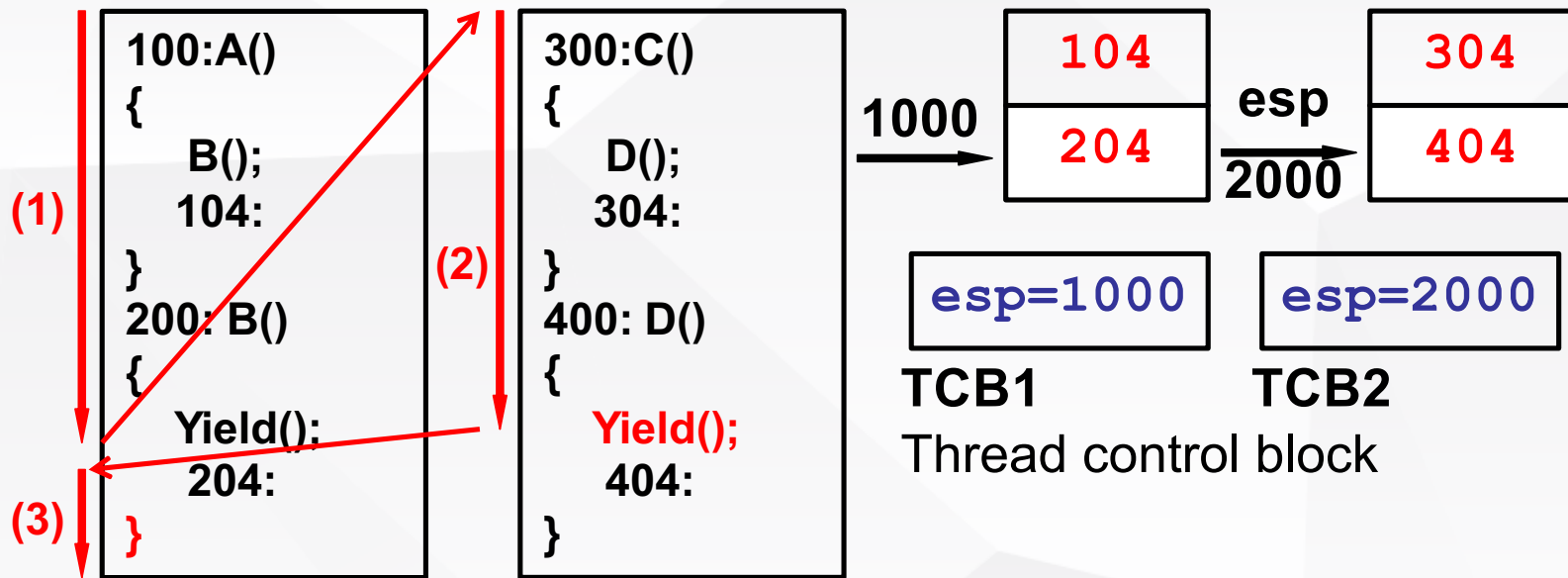
为什么?

共用一个栈会出问题





从一个栈到两个栈...



■ **Yield**切换要先切换栈，然后...

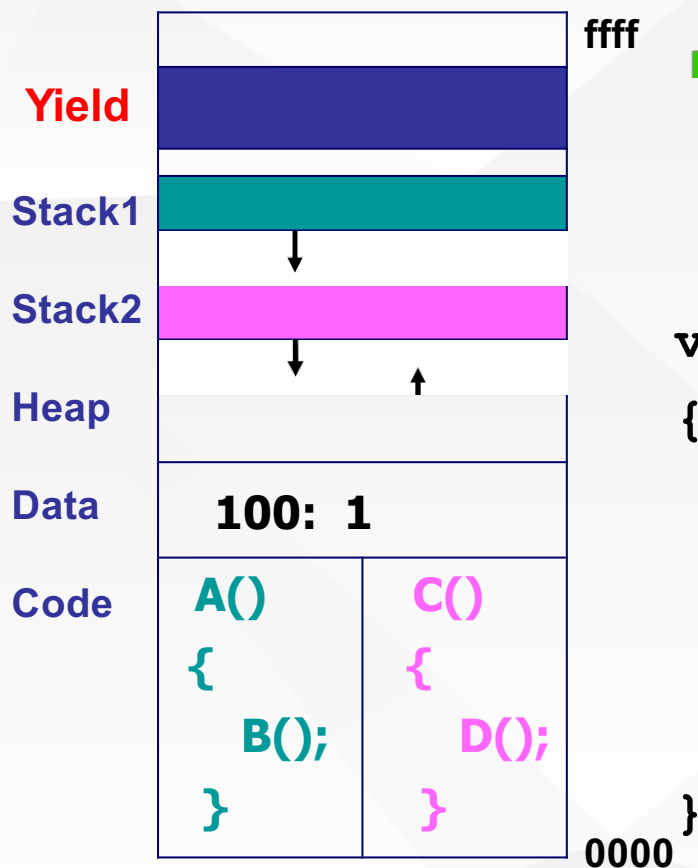
```
void Yield() {  
    TCB2.esp=esp;  
    esp=TCB1.esp;  
    jmp 204; 应该去掉  
}
```

■ (3)再往下执行会怎么样？

■ 204是调用 `Yield()` 才压栈的...



两个线程的样子：两个TCB、两个栈、切换的PC在栈中

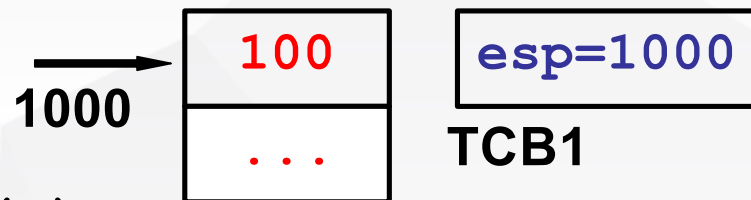


两个执行序列

■ ThreadCreate的核心是实现三样东西：TCB、栈、PC

```
void ThreadCreate(A)
{
```

```
    TCB *tcb=malloc();
    *stack=malloc();
    *stack = A;//100
    tcb.esp=stack;
}
```





将所有东西组合在一起.....

```
void WebExplorer() //main()  
{ ThreadCreate(GetData, URL, buffer); ...  
  while(1) Yield(); }
```

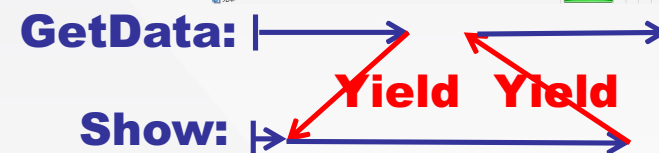
```
void GetData(char *URL, char *p) {  
  连接URL; 下载; Yield(); ... }
```

```
void ThreadCreate(func, arg1) {  
  申请栈; 申请TCB; func等入栈; 关联TCB与栈; ... }
```

```
void Yield() { 压入现场; esp放在当前TCB  
               中; Next(); 从下个TCB取出esp; 弹栈切换线程; }
```

调度函数，对系统影响很大，如可优先调度show!

■ **gcc -o explorer get.c yield.c ... 或 gcc get.c.. -lthread**



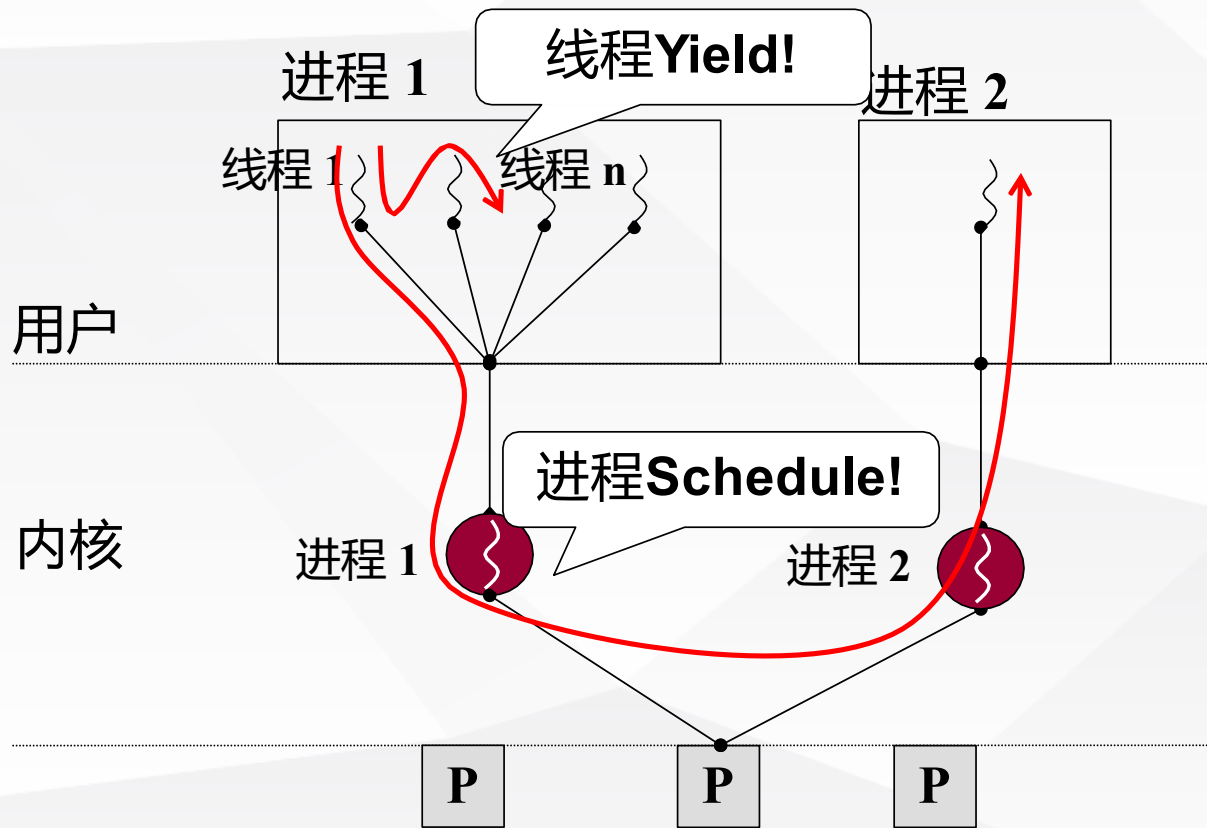
■ **GetData**下载到文本时会调用**Yield()**...





为什么说用户级线程——Yield是用户程序

■ 如果进程的某个线程进入内核并阻塞，则...



GetData

连接URL发起请求;

等待网卡IO...

进程阻塞

Show

显示文本和链接;

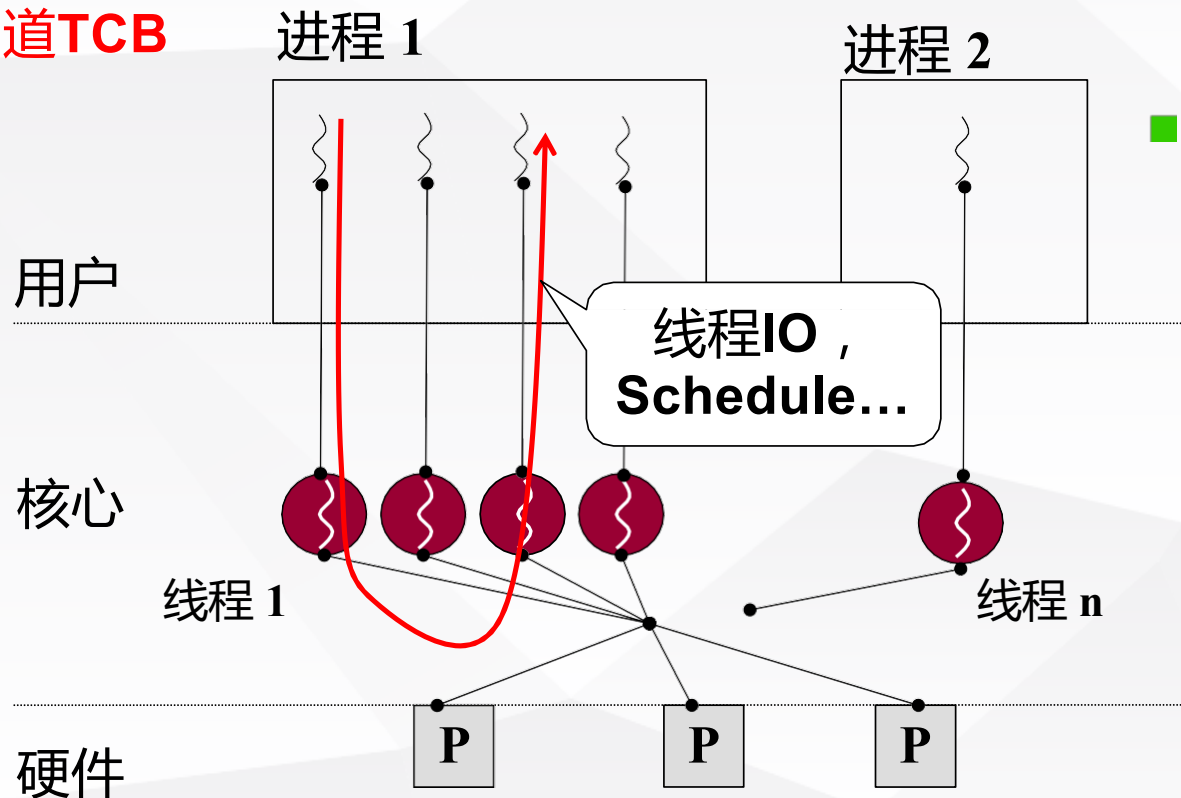
...;



内核级线程



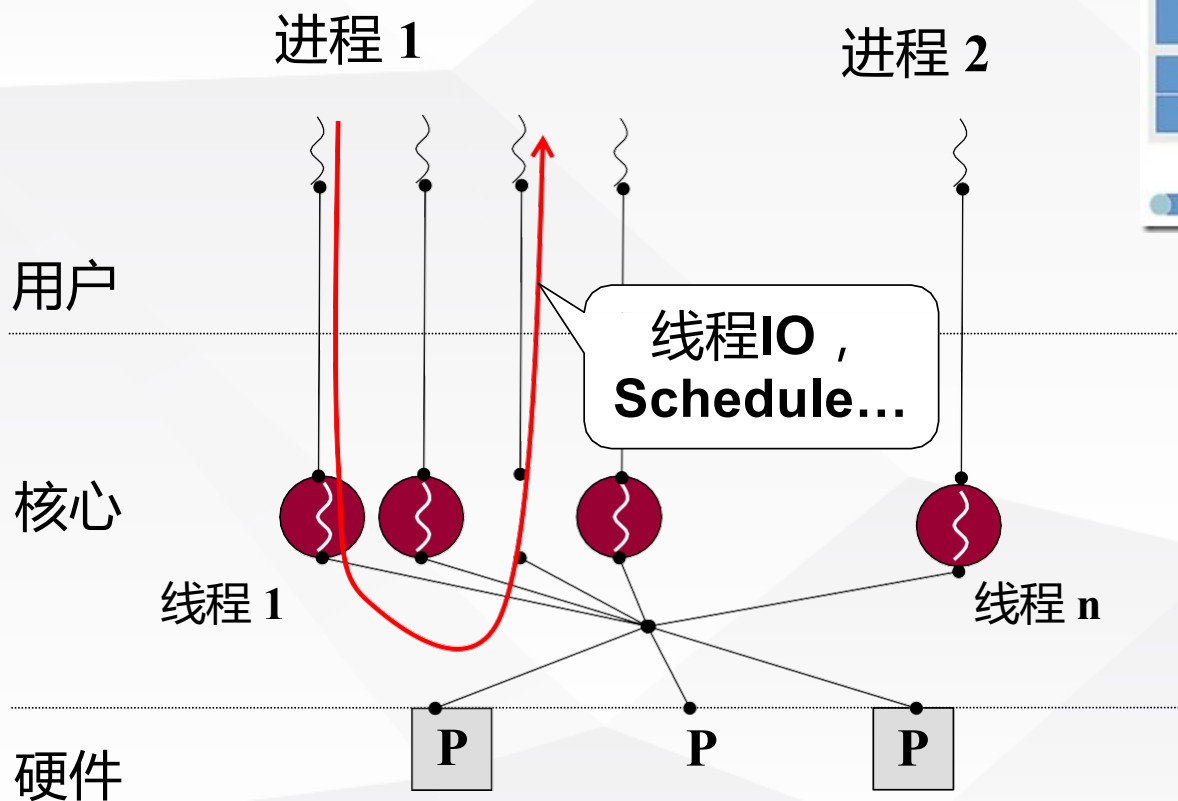
- **ThreadCreate**是系统调用，会进入内核，内核知道TCB



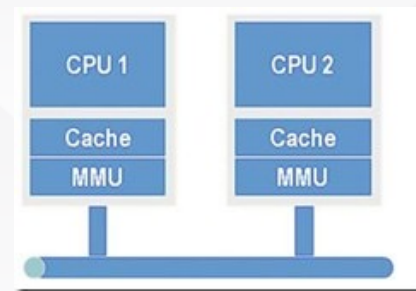
- `gcc -o explorer explorer.c yield.c ...`
- 内核级线程`gcc -o explorer explorer.c...` ; `ThreadCreate`是系统调用；`Yield()`用户不可见，调度点由系统决定，`schedule()`



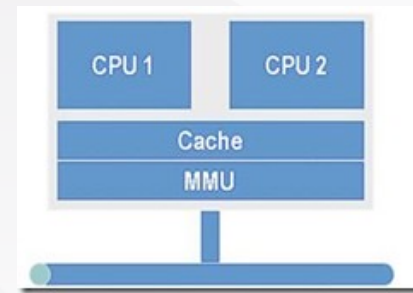
开始内核级线程



多处理器



多核



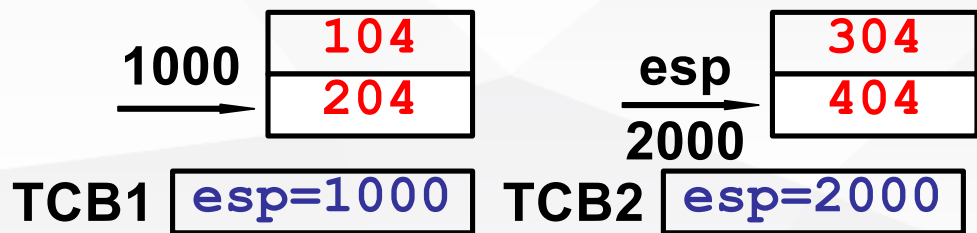
MMU :
memory
management
unit
映射

内核级线程对多核的支持怎么样？



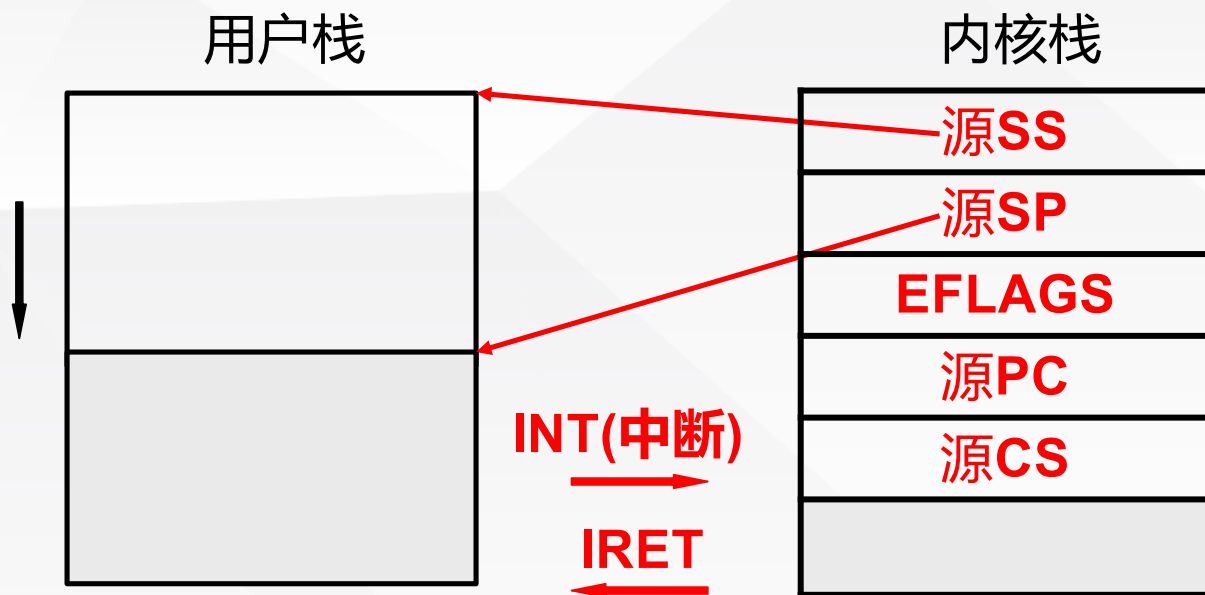
和用户级相比，内核级线程有什么不同？

- **ThreadCreate**是系统调用，内核管理TCB，内核负责切换线程
- 如何让切换成型？ – 内核栈，TCB
 - 用户栈是否还要用？执行的代码仍然在用户态，还要进行函数调用
 - 一个栈到一套栈；两个栈到两套栈
 - TCB关联内核栈，那用户栈怎么办？





用户栈和内核栈之间的关联



■ 所有中断(时钟、外设、INT指令)都引起上述切换

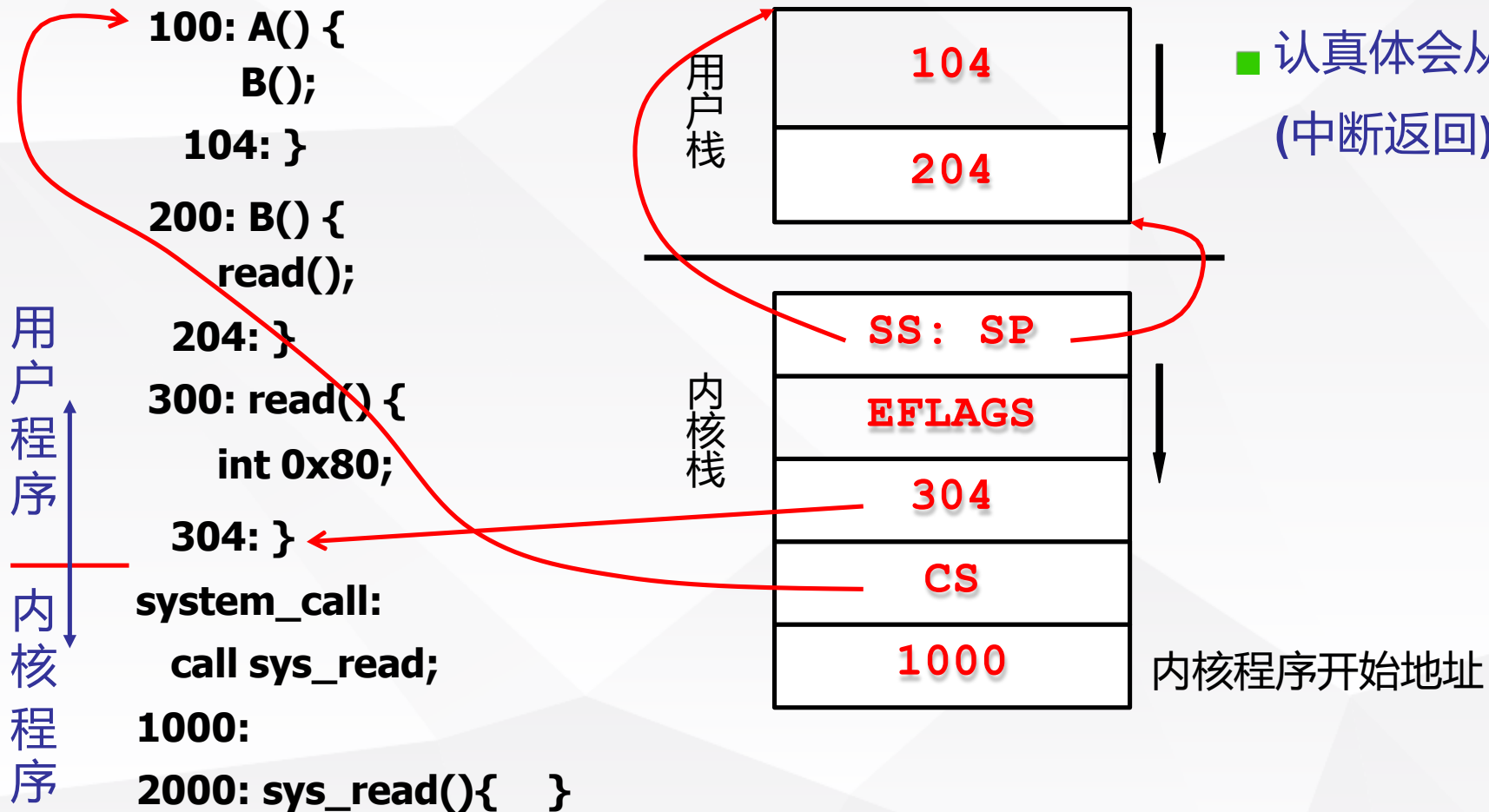
■ 中断(硬件)又一次帮助了操作系统...

源SS、源SP：指向用户栈的位置，用于定位用户栈

源PC、源CS：指向用户级线程代码的位置，用于定位用户级线程代码

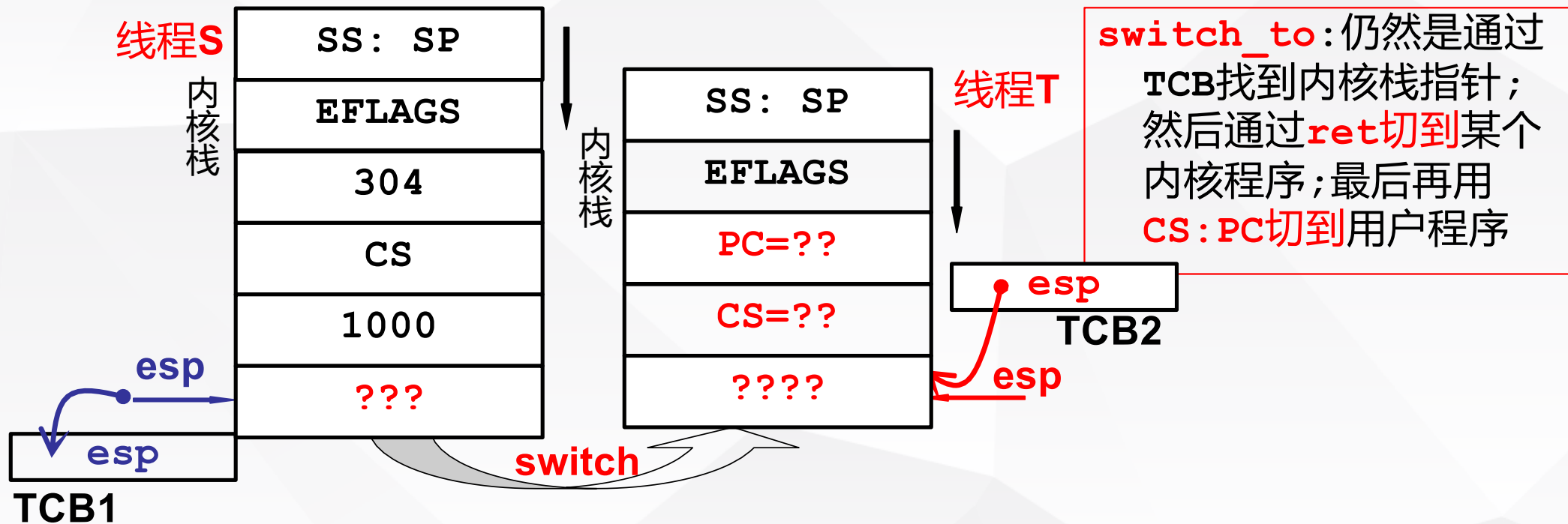


仍然是那个A() , B() , C() , D()...





开始内核中的切换：switch_to



```
sys_read() {启动磁盘读;将自己变成阻塞;找到next;switch_to(cur, next);}
```



内核线程switch_to的五段论

第一级切换

中断入口: (进入切换)

```
push ds; ... pusha;  
mov ds, 内核段号; ...  
call 中断处理
```

??:

中断处理: (引发切换)

```
启动磁盘读或时钟中断;  
schedule();  
} //ret
```

```
schedule: next=..;  
call switch_to;  
} //ret
```

switch_to: (内核栈切换)

```
TCB[cur].esp=%esp;  
%esp=TCB[next].esp;  
ret
```

中断出口: (第二级切换)

```
popa; ...; pop ds;  
iret
```

S、T非同一进程: (地址切换)

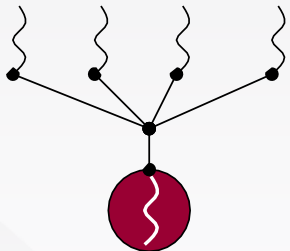
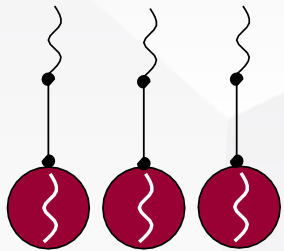
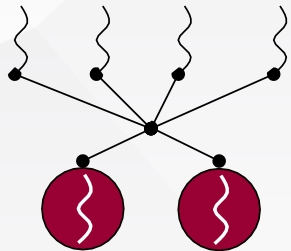
要首先切换地址映射表;

```
TCB[cur].ldtr=%ldtr  
%ldtr=TCB[next].ldtr  
//内存管理
```





用户级线程、内核级线程的对比

	用户级线程	核心级线程	用户+核心级
实现模型			
利用多核	差	好	好
并发度	低	高	高
代价	小	大	中
内核改动	无	大	大
用户灵活性	大	小	大