上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

L3-2. 多线程

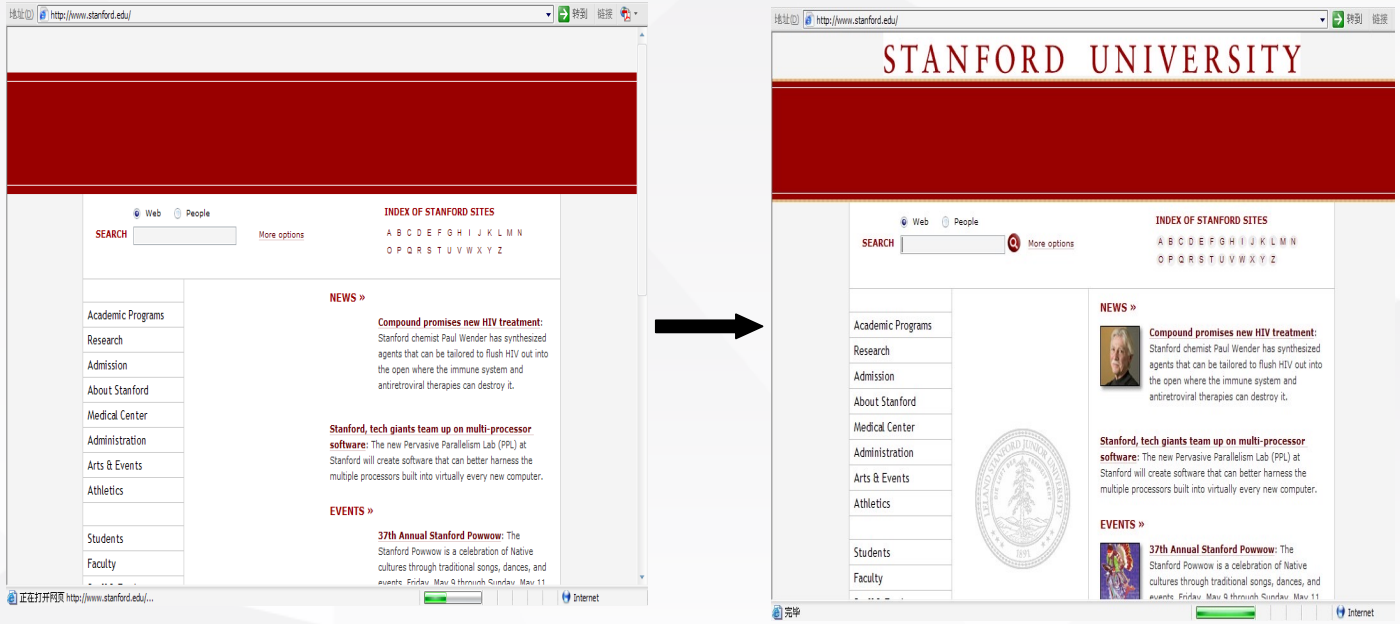
宋卓然

上海交通大学计算机系

songzhuoran@sjtu.edu.cn

多线程的实例（浏览器）

- 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

避免了数据在内存中不必要的移动、拷贝！

使用pthread实现浏览器

```
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){...};
```

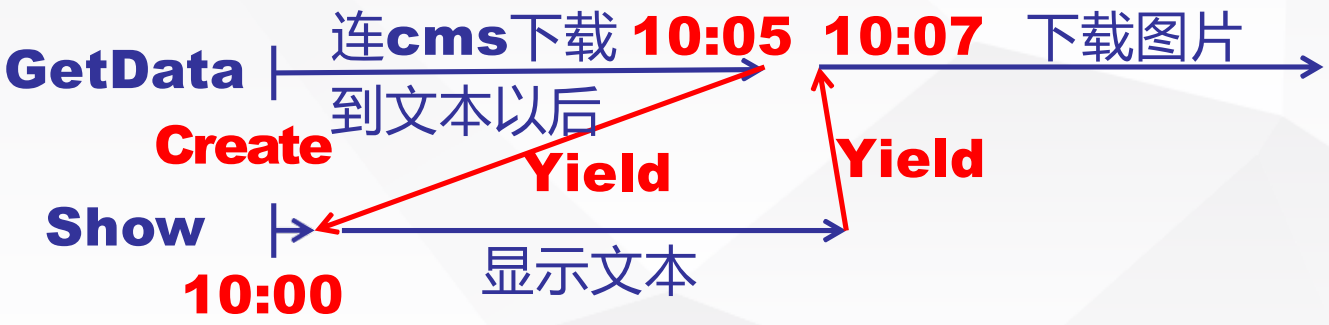
Yield如何实现

样子弄明白了，剩下的就是写程序实现这个样子了…

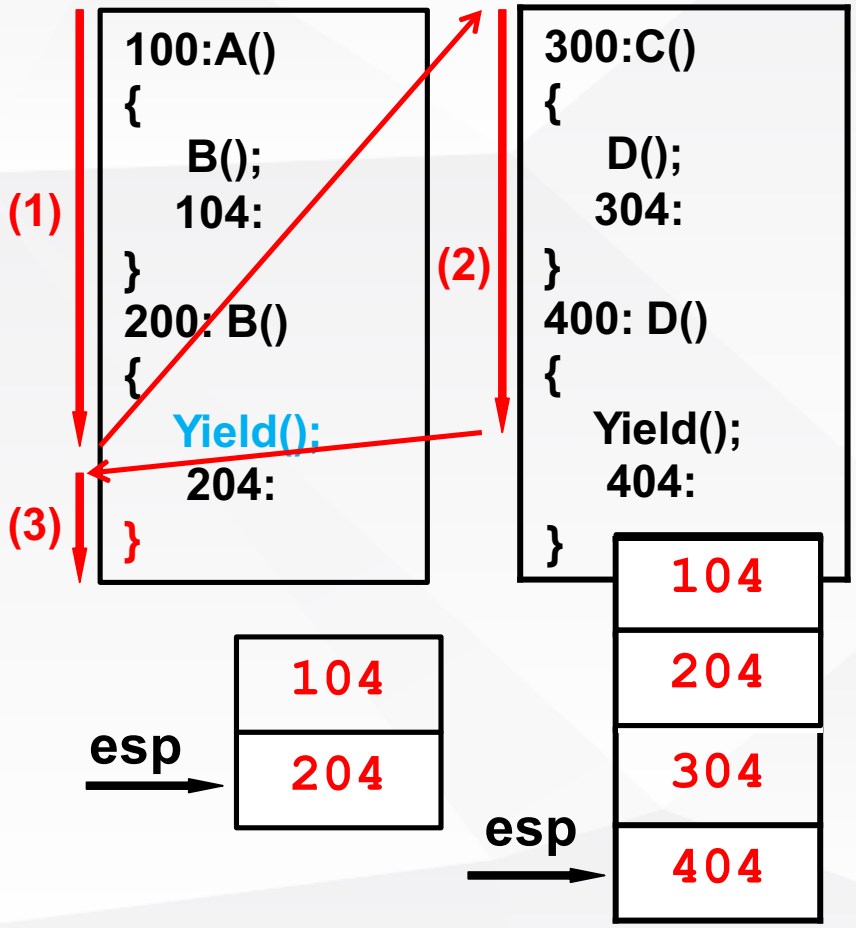
- 核心是Yield…
  - 切换线程
- 仔细看Yield，就是100跳到300
  - 切换程序

```
100:A()
{
    B();
    104:
}
200: B()
{
    Yield();
    204:
}
```

```
300:C()
{
    D();
    304:
}
400: D()
{
    Yield();
    404:
}
```



两段代码与一个栈




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

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

- *(3)*再往下执行会怎么样*?*

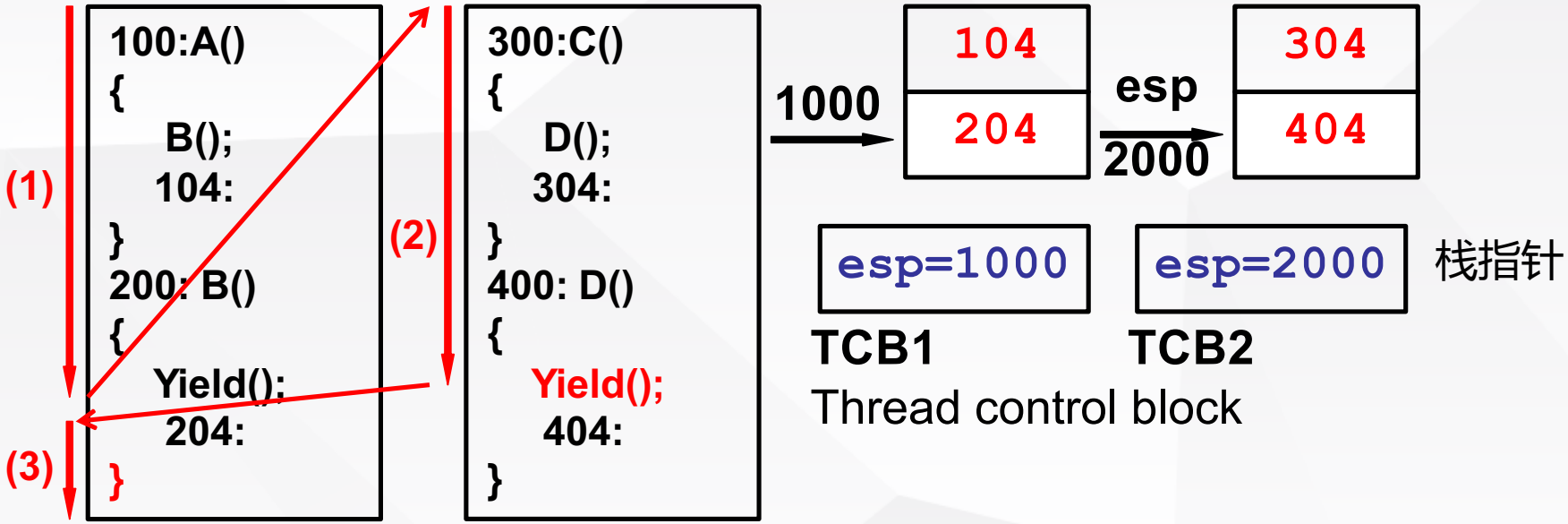
  花括号弹栈

- 问题怎么解决?

  为什么?

  共用一个栈会出问题

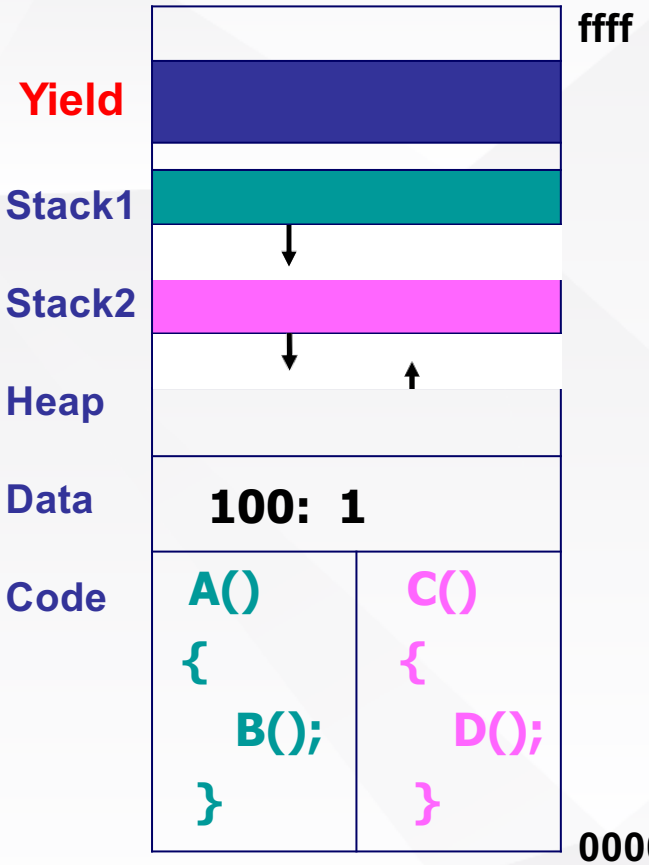
从一个栈到两个栈



- Yield切换要先切换栈，然后...

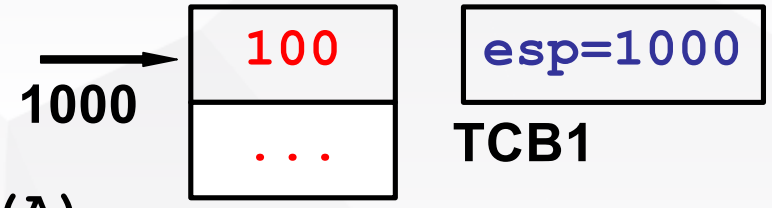
```
void Yield(){
    TCB2.esp=esp;
    esp=TCB1.esp;
}
```

两个线程的切换：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（栈内返回的地址）




```
void ThreadCreate(A)
{
    TCB *tcb=malloc();
    *stack=malloc();
    *stack = A;//100 函数的起始地址
    tcb.esp=stack;
}
```

将所有的东西组合在一起……

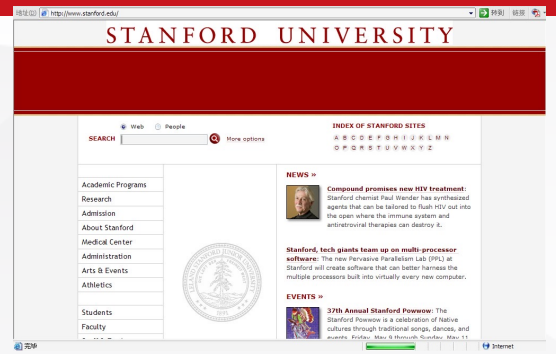
```
void WebExplorer()//main()
{ ThreadCreate(GetData,URL, buffer);...
  while(1)Yield(); }
```

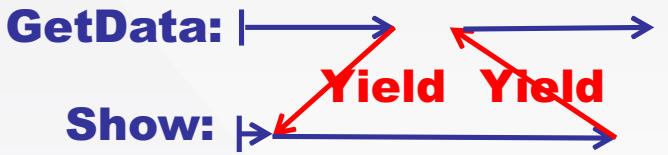
```
void GetData(char *URL, char *p){
  连接URL;下载; Yield();...}
```

```
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!



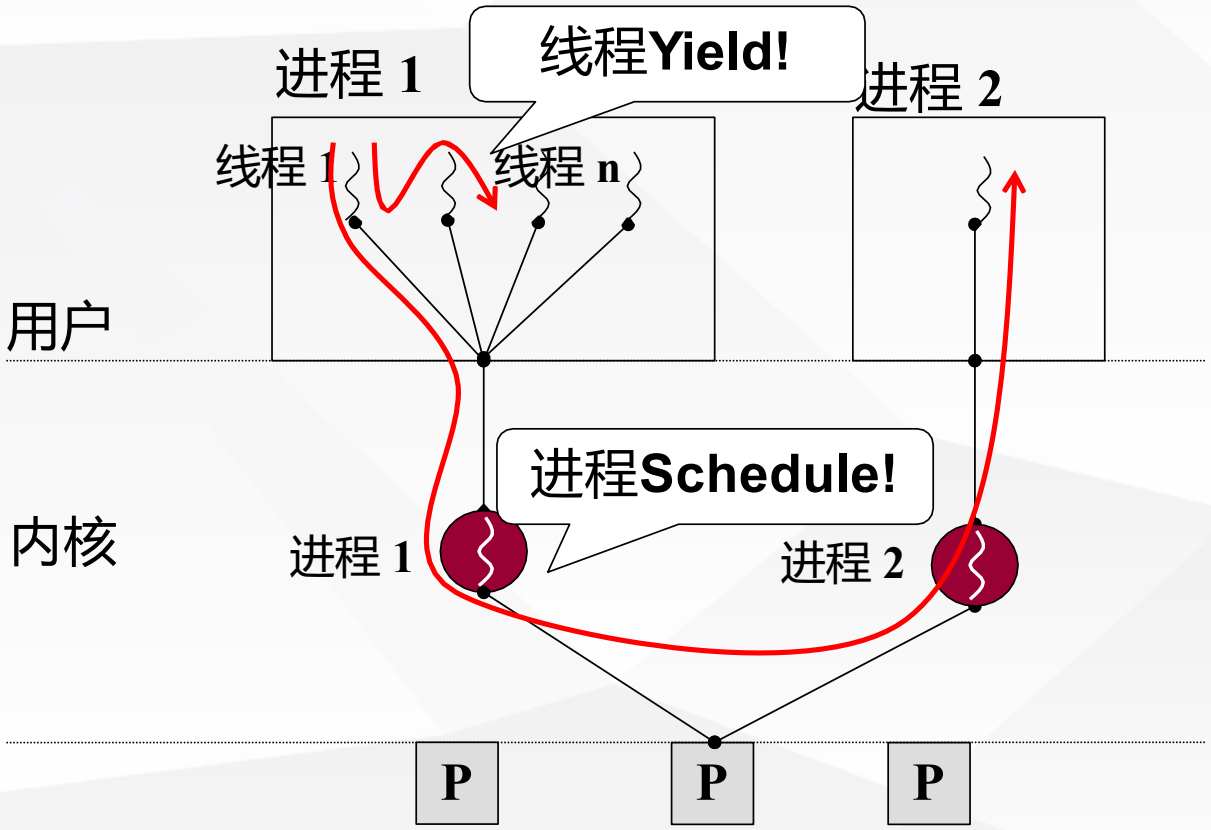


- GetData下载到文本时会调用Yield()…

- gcc -o explorer get.c yield.c … 或 gcc get.c.. -lthread

为什么说是用户级线程——Yield是用户程序

- 如果进程的某个线程进入内核并阻塞，则…



GetData

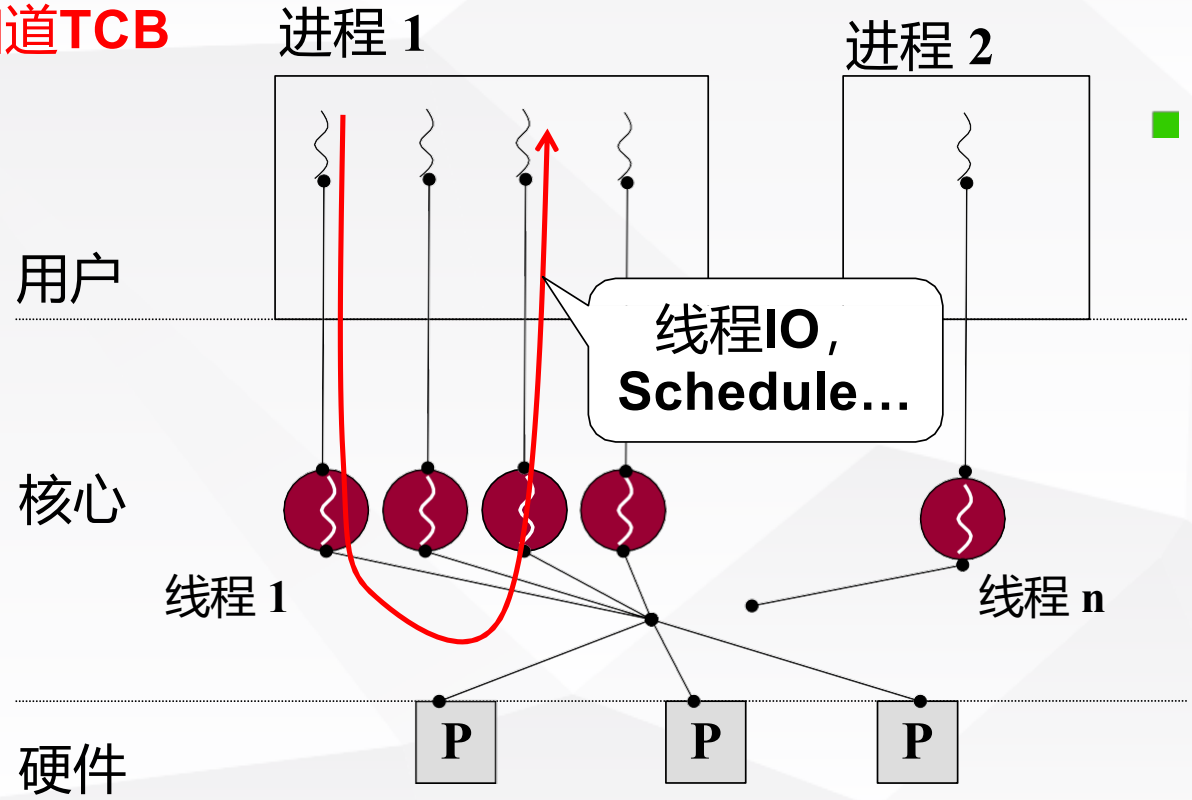
```
连接URL发起请求；
等待网卡IO...
进程阻塞
```

Show 内核看不到show函数

```
显示文本和链接；
...;
```

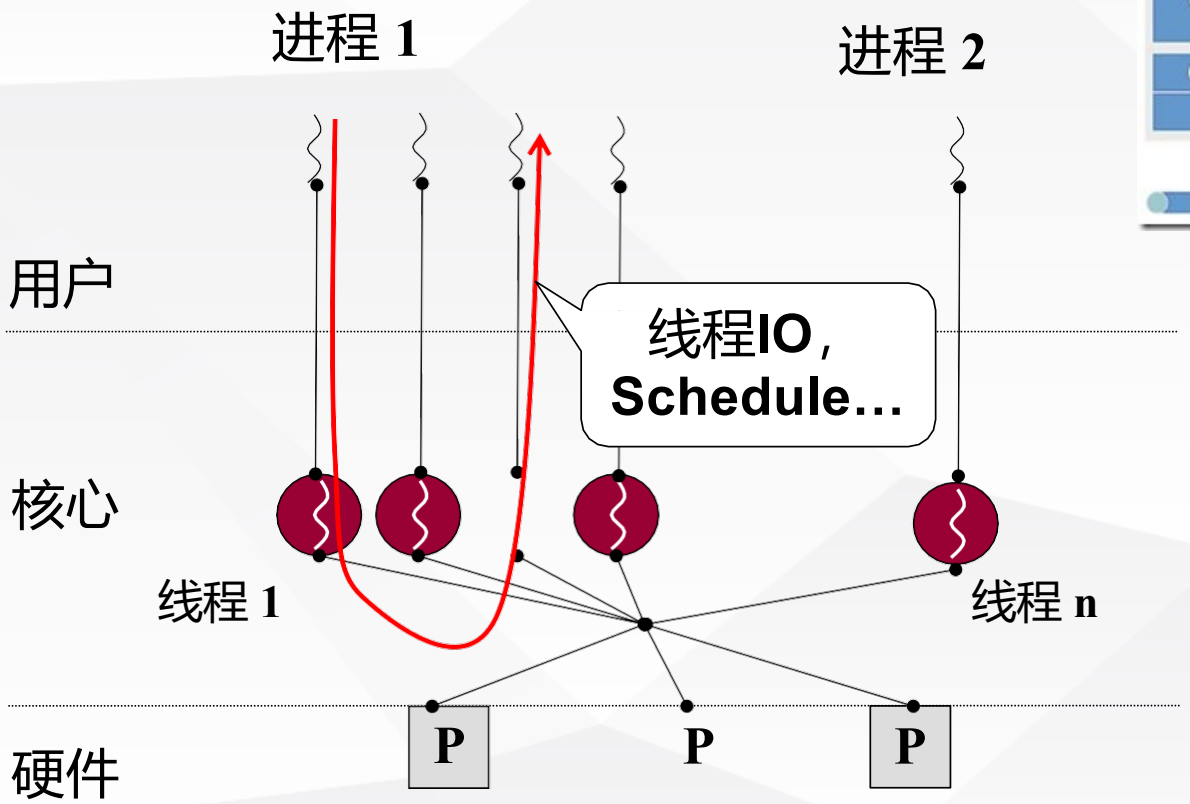
内核级线程

- ThreadCreate是系统调用，会进入内核，内核知道TCB

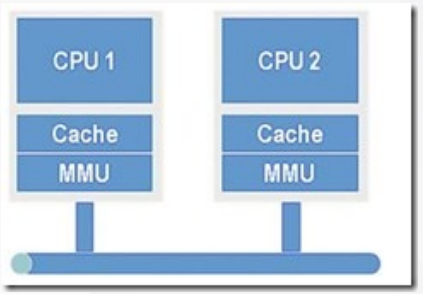


- gcc -o explorer explorer.c yield.c …
- 内核级线程gcc -o explorer explorer.c…；ThreadCreate是系统调用；Schedule()用户不可见，调度由操作系统决定

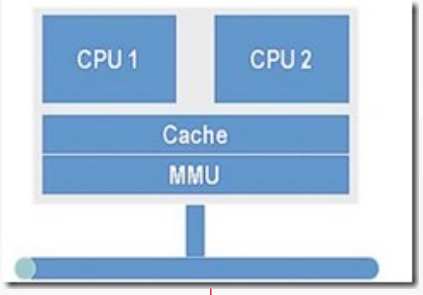
内核级线程



多处理器



多核（共享一个MMU，对应多线程）

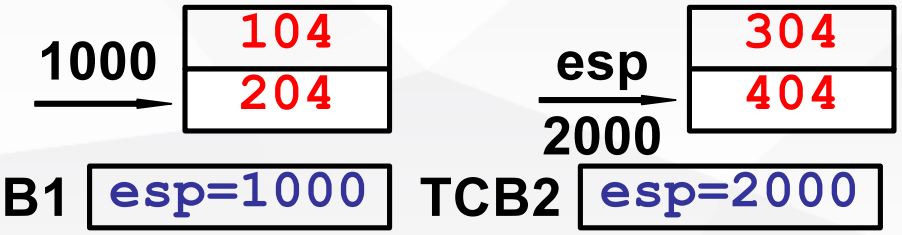


MMU：memory management unit 映射

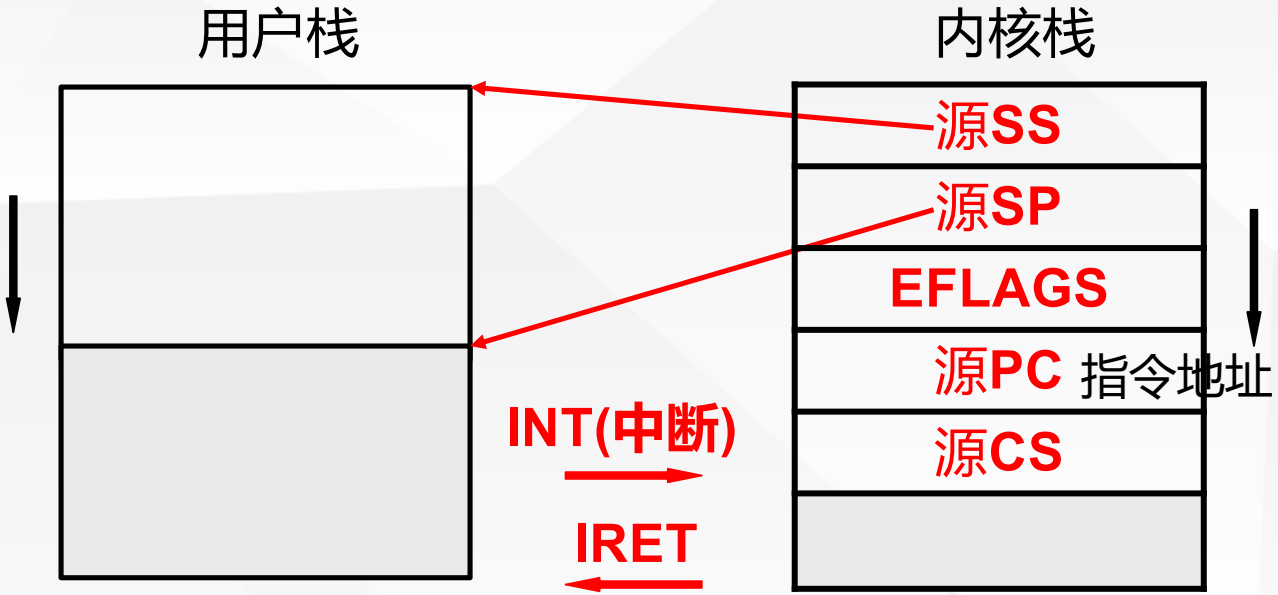
一套MMU的情况下，只有多进程而无多线程的情况下无法提高并行度！

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?

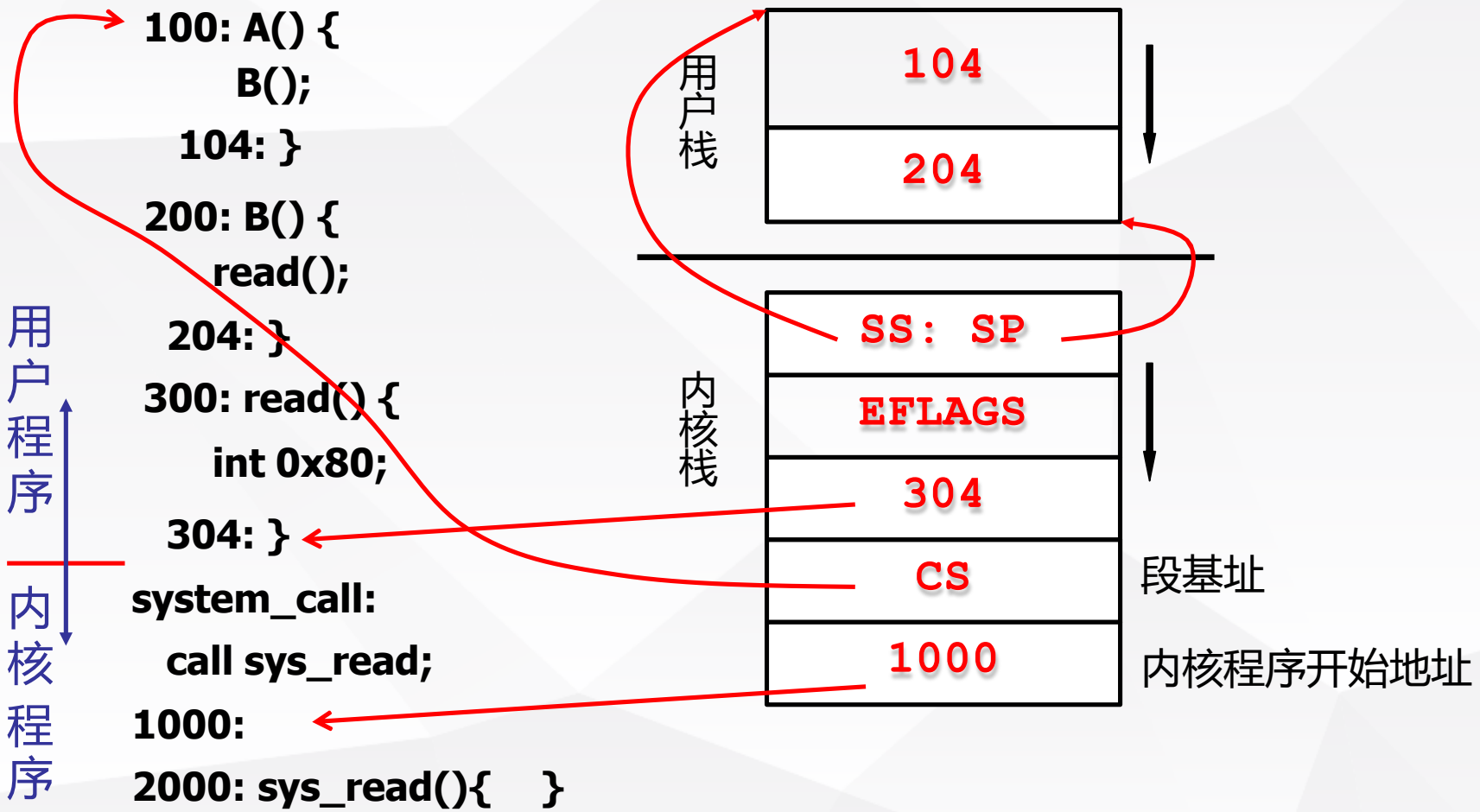


用户栈和内核栈之间的关联

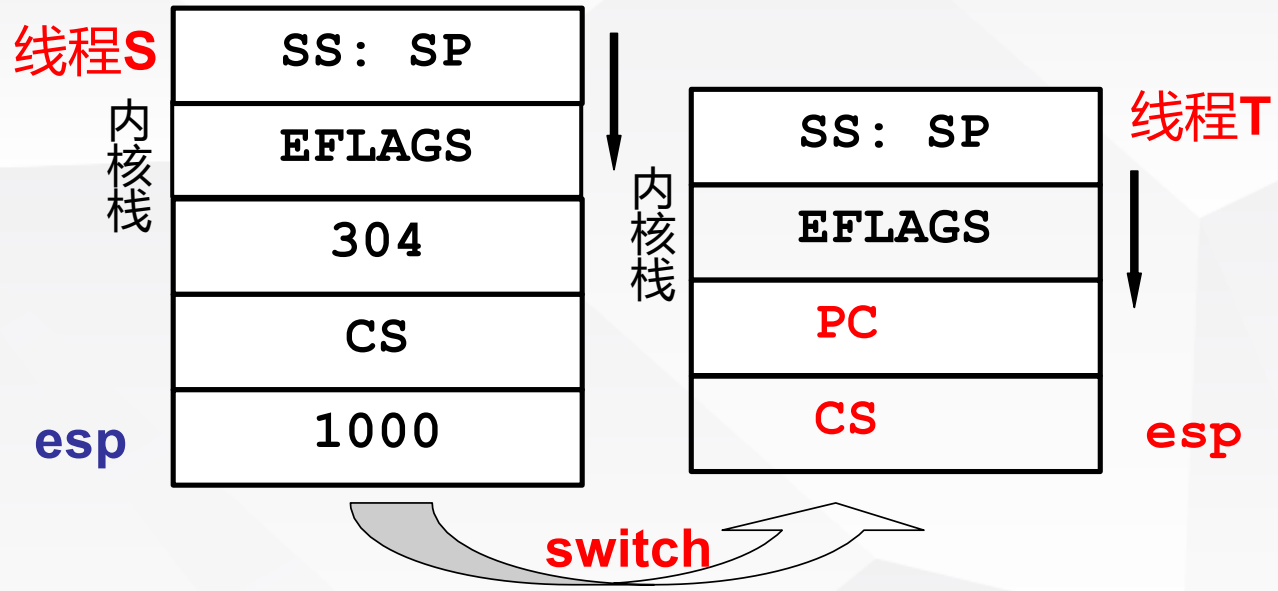


- 所有中断(时钟、外设、INT指令)都引起上述切换
- 中断(硬件)又一次帮助了操作系统

仍然是那个A()，B()，C()，D()...


100: A() {
B();
104: }
200: B() {
read();
204: }
300: read() {
int 0x80;
304: }
system_call:
call sys_read;
1000:
2000: sys_read(){ }
用户程序
内核程序
用户栈
104
204
内核栈
SS: SP
EFLAGS
304
CS
1000
段基址
内核程序开始地址

开始内核中的切换：switch_to



switch_to：仍然是通过TCB找到内核栈指针；然后通过ret切到某个内核程序；最后再用CS:PC切到用户程序

TCB1

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
```