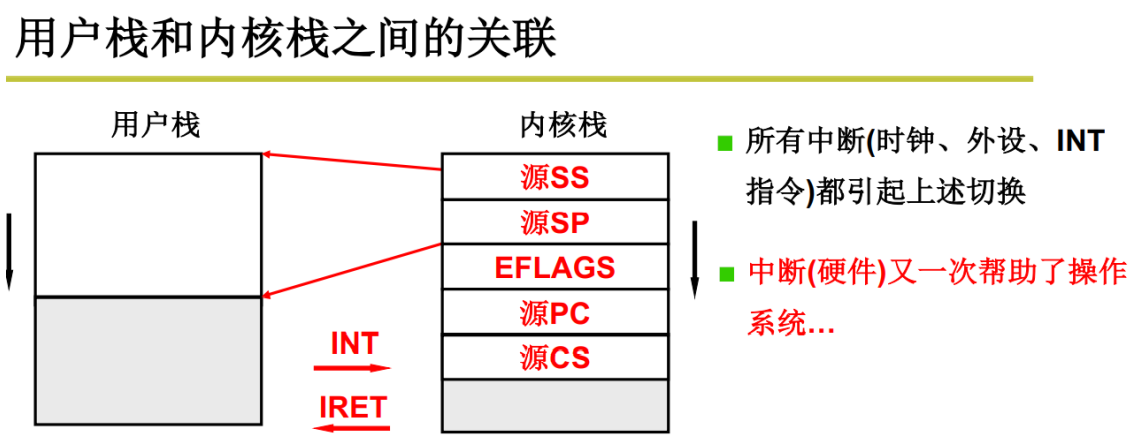
# 哈工大操作系统-L11内核级线程



- **没有用户级进程，进程都是需要访问内核的。**
- **线程可以分用户级和内核级。**理解用户级线程对理解内核级线程有很大帮助。
- 多内核级线程可以利用充分利用多核CPU的特性。(进程不行，用户级线程也不行)

## 1.和用户级相比，核心级线程有什么不同?

- 需要两套栈，内核栈和用户栈
  - 因为内核级线程由系统调用创建，由内核负责切换，因此会有存在内核中的内核栈
  - 而代码依然是用户态中执行的，因此也要有一个用户栈
- 内核级线程切换的时候，内核栈完成切换的同时用户栈也要进行切换

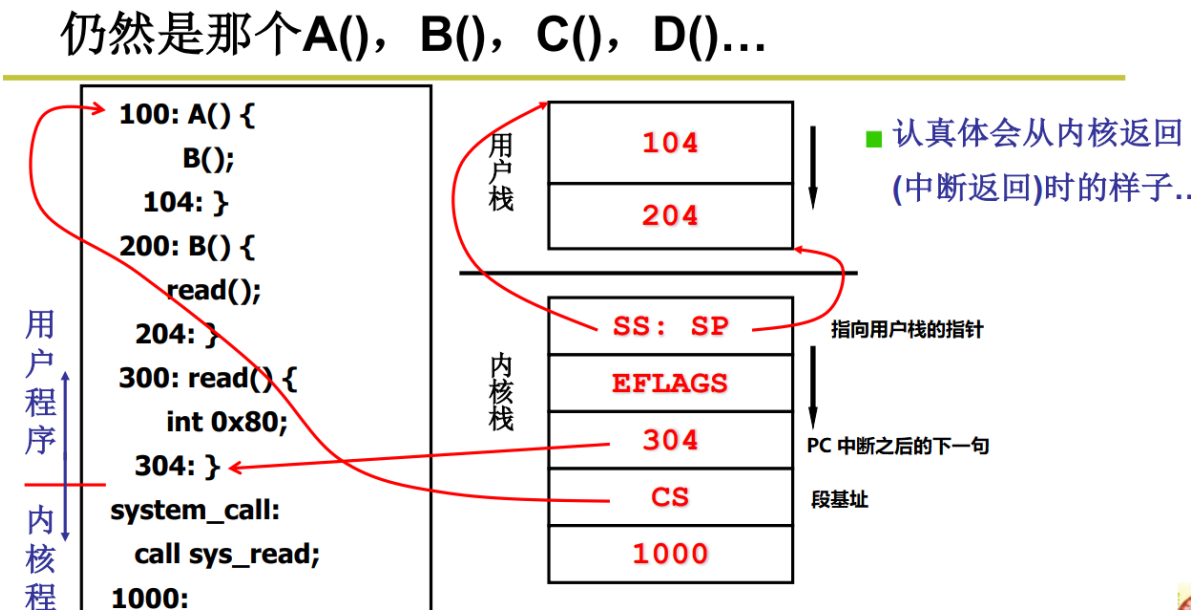



即从上图可以知道，在中断开始后，内核栈会保存指向用户栈的指针，以及其他必要信息。

## 2.内核级线程如何切换

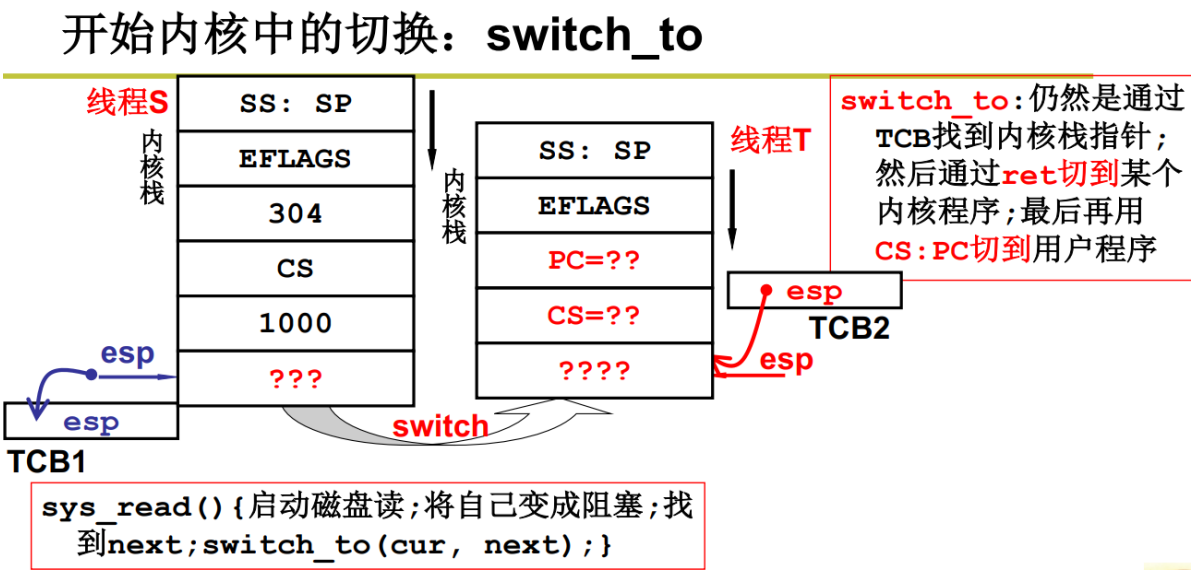
- 真正要执行的大多都是用户的程序，但**两个内核级线程切换需要在内核走一遭。**
- 通过系统支持的内核线程切换，**从用户到内核到内核再到用户。**
- **找到内核级线程TCB需要进入内核**

## 2.1 内核线程A的用户程序-转向-内核线程A的内核程序:用户栈到内核栈的转换




- 执行到300的时候，INT中断后，注意栈的切换动作。
- 执行完中断处理的所有程序之后(即1000那些也执行完毕)，从中断返回的时候，会弹出CS和PC，即返回到304处继续下面的执行。而内核栈上方的东西是帮助寻找用户栈的。

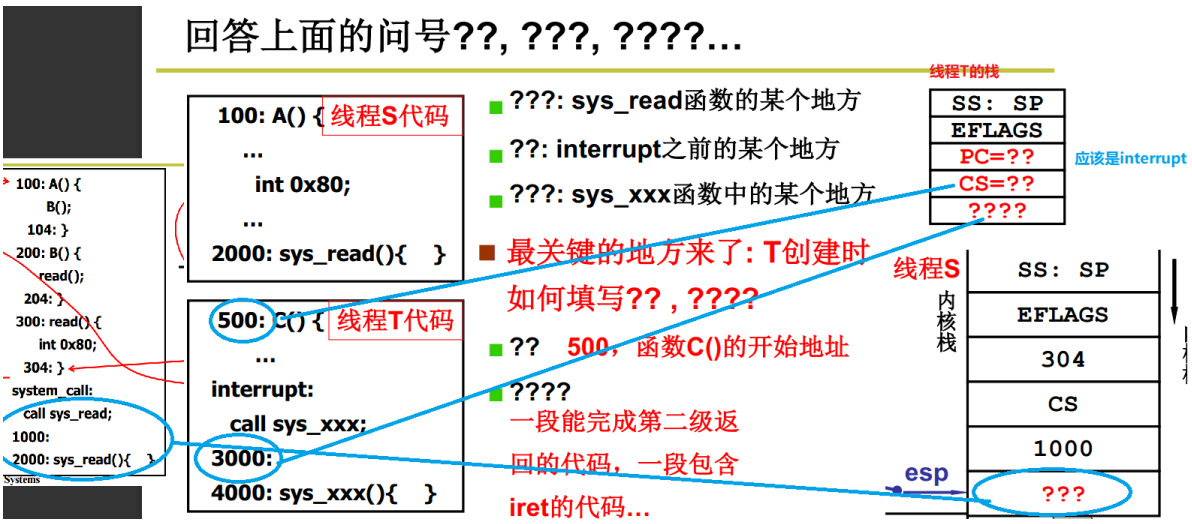
## 2.2 内核线程A内核程序-转向-内核线程B的内核程序:切换内核栈




- 切换前，先将当前的esp即当前的栈地址等信息保存在当前线程的TCB中
- 然后，通过切换目标内核线程的TCB知道目标的内核栈的指针，切换到另一个内核线程

## 2.3 内核线程B内核程序-转向-内核线程B的用户程序

从B线程的内核程序，通过内核栈中保存的CS和PC切回到B线程的用户程序




## 2.4 内核级线程切换的五段论

### 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();引发切换
}//ret
```

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

找到下一个东西的TCB然后进入内核完成TCB的切换等

switch_to:(内核栈切换)

```
TCB[cur].esp=%esp;
%esp=TCB[next].esp;
ret 当前执行的内核栈切换成另一个线程的
```

中断出口:(第二级切换)

```
popa;...; pop ds;
iret 中断返回, 从另一个线程的内核栈返回到另一个线程的用户程序
```

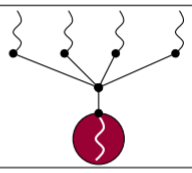
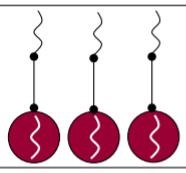
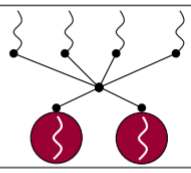
S、T非同一进程:(地址切换)

```
要首先切换地址映射表;
TCB[cur].ldtr=%ldtr
%ldtr=TCB[next].ldtr
//内存管理
```

进程切换多一个地址映射表的切换


## 3.用户级线程和内核级线程的对比

### 用户级线程、核心级线程的对比

| | 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 |