**[用户态和核心态的区别](https://www.cnblogs.com/Allen-rg/p/7171105.html)**

**1.操作系统需要两种CPU状态：**

**内核态（Kernel Mode）：运行操作系统程序**

**用户态（User Mode）：运行用户程序**

**2.指令划分：**

特权指令：**只能由操作系统使用、用户程序不能使用的指令。**  举例：启动I/O   内存清零  修改程序状态字  设置时钟    允许/禁止终端   停机

非特权指令：用户程序可以使用的指令。  举例：控制转移  算数运算  取数指令   访管指令（使用户程序从用户态陷入内核态）

**3.特权级别：**

**特权环：R0、R1、R2和R3**

**R0相当于内核态，R3相当于用户态；**

**不同级别能够运行不同的指令集合；**

**4.CPU状态之间的转换：**

**用户态--->内核态：唯一途径是通过中断、异常、陷入机制（访管指令）**

**内核态--->用户态：设置程序状态字PSW**

**5.内核态与用户态的区别：**

**1）内核态与用户态是操作系统的两种运行级别，当程序运行在3级特权级上时，就可以称之为运行在用户态。**因为这是最低特权级，是普通的用户进程运行的特权级，**大部分用户直接面对的程序都是运行在用户态；**

**2）当程序运行在0级特权级上时，就可以称之为运行在内核态。**

**3）运行在用户态下的程序不能直接访问操作系统内核数据结构和程序。当我们在系统中执行一个程序时，大部分时间是运行在用户态下的，在其需要操作系统帮助完成某些它没有权力和能力完成的工作时就会切换到内核态。**

**4）这两种状态的主要差别是**：

处于用户态执行时，**进程所能访问的内存空间和对象受到限制，**其所处于占有的处理机是可被抢占的 ；

而处于核心态执行中的进程，**则能访问所有的内存空间和对象，**且所占有的处理机是不允许被抢占的。

**6. 通常来说，以下三种情况会导致用户态到内核态的切换：**

**1）系统调用**

**这是用户态进程主动要求切换到内核态的一种方式**，**用户态进程通过系统调用申请使用操作系统提供的服务程序完成工作。**比如前例中**fork()**实际上就是**执行了一个创建新进程的系统调用。**

而**系统调用的机制**其核心还是**使用了操作系统为用户特别开放的一个中断来实现**，例如Linux的int 80h中断。

**2）异常**

当**CPU在执行运行在用户态下的程序时，发生了某些事先不可知的异常**，这时会触发由当前运行进程**切换到处理此异常的内核相关程序中**，也就转到了内核态，比如缺页异常。

**3）外围设备的中断**

**当外围设备完成用户请求的操作后，会向CPU发出相应的中断信号**，这时CPU会暂停执行下一条即将要执行的指令转而去执行与中断信号对应的处理程序，

如果先前执行的指令是用户态下的程序，那么这个转换的过程自然也就发生了由用户态到内核态的切换。比如**硬盘读写操作完成**，系统会**切换到硬盘读写的中断处理程序中执行后续操作等。**

**这3种方式是系统在运行时由用户态转到内核态的最主要方式，其中系统调用可以认为是用户进程主动发起的，异常和外围设备中断则是被动的。**

2）具体的切换操作

从触发方式上看，可以认为存在前述3种不同的类型，但是从**最终实际完成由用户态到内核态的切换操作上来说**，涉及的关键步骤是完全一致的，没有任何区别，都**相当于执行了一个中断响应的过程，因为系统调用实际上最终是中断机制实现的**，而**异常和中断的处理机制基本上也是一致的**，关于它们的具体区别这里不再赘述。关于中断处理机制的细节和步骤这里也不做过多分析，涉及到由用户态切换到内核态的步骤主要包括：

[1] 从当前进程的描述符中提取其内核栈的ss0及esp0信息。

[2] 使用ss0和esp0指向的内核栈将当前进程的cs,eip,eflags,ss,esp信息保存起来，这个

过程也完成了由用户栈到内核栈的切换过程，同时保存了被暂停执行的程序的下一

条指令。

[3] 将先前由中断向量检索得到的中断处理程序的cs,eip信息装入相应的寄存器，开始

执行中断处理程序，这时就转到了内核态的程序执行了。