内核态与用户态

C-2 创建: 张林伟, 最后修改: 张林伟 01-18 11:54

内核态（CPU特权模式）和用户态（CPU用户模式）

- 防止应用程序进行越权的操作
- 防止应用程序越权访问内存时使用了虚拟地址空间映射的技术，这是操作系统软件配合硬件的MMU共同实现的。在用户模式下，应用程序访问的内存地址是虚拟内存地址，会映射到操作系统指定的物理地址上。这个虚拟内存地址空间就是你说的用户空间
- 应用程序无法自由进入内核态，只能通过操作系统提供的接口调用进入，或者在硬件中断到来时被动进入
- 应用程序通过操作系统功能来使用硬件
- 运算器是无状态的；控制器配合一部分寄存器，但是寄存器数量很少，而且通常都很容易被修改；输入设备、输出设备只有接受指令的时候才动作。归根结底来说，整个计算机的运行状态几乎完全由存储器和少数几个寄存器控制
- 物理内存就是整个计算机状态的全部，如果程序有办法读写所有的物理内存和寄存器，那任何保护手段都无济于事。所以要限制应用程序的行为，必须在应用程序和操作系统执行时有不同的状态，核心问题在于保护关键寄存器和重要的物理内存

> 现代MMU通常使用虚拟地址空间的技术来解决这个问题，也就是你说的“用户空间”。在用户模式下，所有访问内存的地址实际上都是**虚拟地址**，它与实际的物理地址是对应不上的。这样，即便两个应用程序使用了相同的地址，它们也可以做到互不干扰，只需要通过技术手段让它们实际映射到不同的物理地址就行了。MMU和操作系统通过称作页表的数据结构来实现虚拟地址到物理地址的映射，一般来说在x86-64系统中，内存按照4KB的大小分成页，每个地址对齐的页可以独立从任意一个虚拟地址段，映射到任意一个物理内存地址段，两个起始地址的低12位都是0（也就是所谓地址对齐，这样任意一个虚拟地址映射到物理地址时，最低12位不需要动）。页表的结构在每次进入用户模式之前都可以重新设置，这样切换进程之后，页表发生了变化，同一个虚拟地址就会映射到不同的物理地址上，这就同时实现了多个目标：
>
> - 应用程序有独立的虚拟地址空间
> - 应用程序只能访问已经映射了的虚拟地址空间，未映射的物理地址无法访问（实现了保护内存）
> - 页表和中断向量表，理所当然不会被映射出来
> - 部分RISC（x86是CISC）的架构上，内存和外部设备有统一的地址空间，不映射外设的地址，也就阻止了对外设的访问
> - 应用程序看来连续的内存，在物理内存上不需要是连续的，内存使用的效率很高
> - 以某些方式访问某些页面时可以触发操作系统的中断，操作系统可以趁这个机会修改页表，这就给操作系统实现高级内存管理功能打下了基础

用户态切换到内核态

- 系统调用
- 异常事件，如缺页异常
- 外围设备的中断

系统调用的本质其实也是中断，相对于外围设备的硬中断，这种中断称为**软中断**。

参考文档：

https://www.zhihu.com/question/306127044/answer/555327651

https://www.jianshu.com/p/85e931636f27