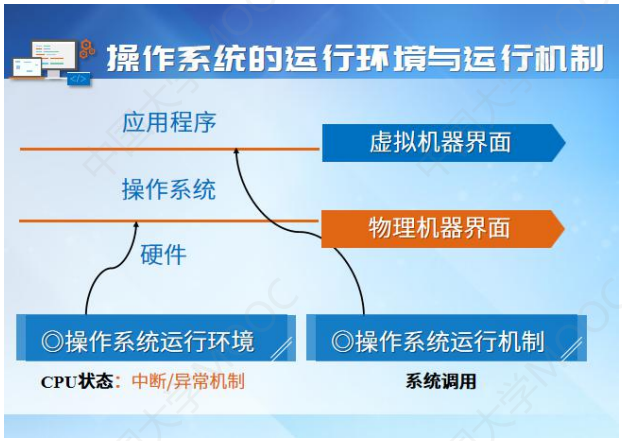
1.5 节课前预习宝典

问题：什么是操作系统的运行环境和运行机制？

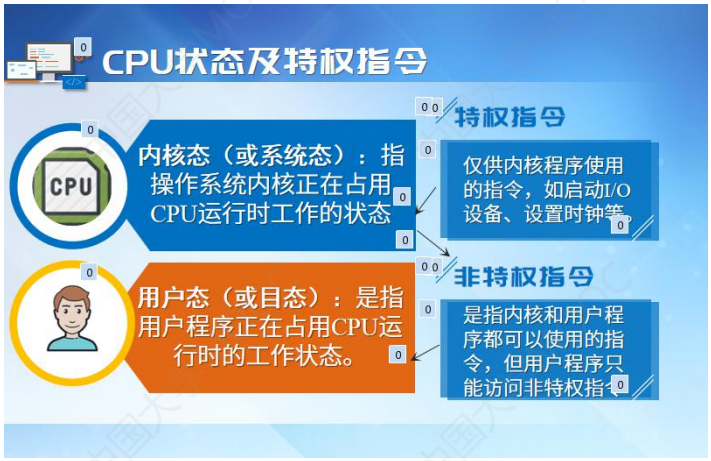
什么是环境，想想你的宿舍环境吧，舍友如果是打游戏的猪友，你要好好学习会被鄙视的。那 OS 的环境是什么？简单的说就是硬件吧，可是硬件的说法太笼统，更具体的说就是 CPU，内存，硬盘等等，有这些就能运行起来？舍友没有网和电脑能玩起游戏？CPU 要运行程序是否是要一组寄存器配合它（存储思想），否则，它要是每个字节都从内存取还不把它累死，有哪些寄存器呢，去看 CPU 的运行现场有那些寄存器吧。且慢，那要敲键盘咋办？CPU 要傻傻的等么，NO，你在敲键盘时，CPU 早都去玩股票去了，为啥可以这样，因为当你把字符敲完后可以向 CPU 发出一个请求信号，CPU 就可以不玩股票了来处理你的请求，这就是中断机制。

如果我想要从磁盘读取数据，或者想让一个进程生个儿子等这些脏活累活该怎么办，总不能让我们每个用户都辛辛苦苦的与硬件打交道吧，且慢，这些活操作系统都扛起来了，你只需要简单的调用一下函数就可以，比如 read()读数据，fork()生孩子，这就是系统调用。



问题： 什么是处理器（CPU）的状态？

其实，CPU 本来是没有什么状态的，就像你不干事的时候既没有兴奋的状态，也没有厌烦的状态，可是，当老师给你布置了任务，你可能就烦了。CPU 也是，当它不执行程序的时候，是没有什么状态的，当它执行的程序是用户程序的时候就叫用户态，当执行的程序是操作系统的代码时就叫系统态或者内核态。



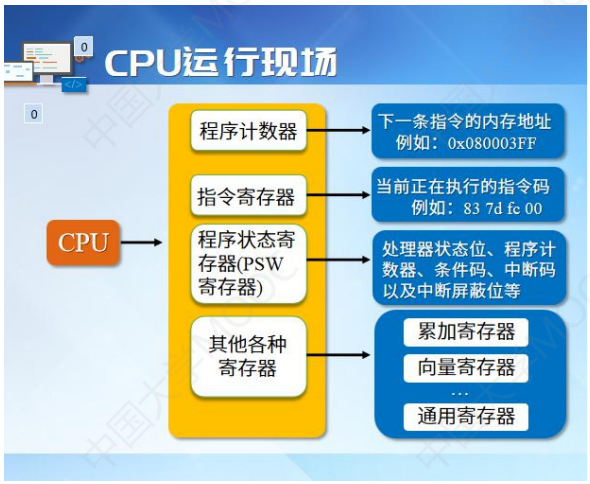
问题：什么是特权指令和非特权指令？

一听特权我们一般都很反感，其实计算机中也有特权，谁有？那就是操作系统。也就是有一部分指令只有操作系统能使用，用户不能使用，这就叫特权指令。而有一部分指令用户态和

操作系统都能使用，这就是非特权指令。（参看 1.5 节）

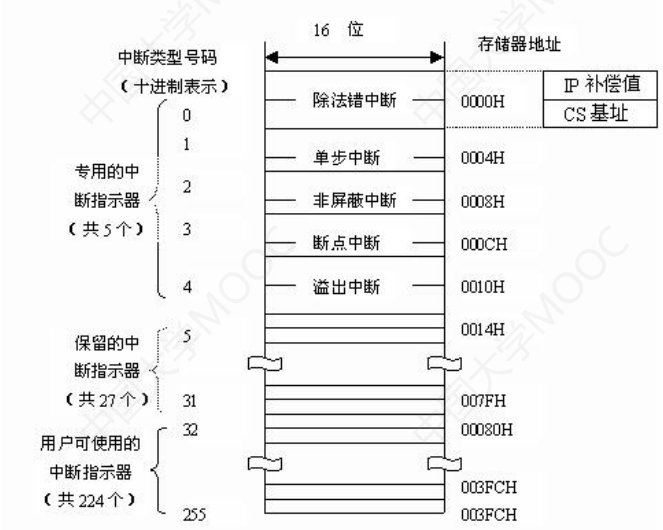
问题：什么是 CPU 运行现场？处理器（CPU）的状态放在哪里？

CPU 运行现场到底是什么？你联想一下车祸现场。CPU 运行现场简单说就是一组寄存器的集合，因为 CPU 在运行过程中无非是一组寄存器与它配合，也即是存放指令和地址的一组寄存器，专业的说就是 CPU 运行现场指在程序的执行过程中任一时刻状态信息的集合。那么 CPU 的状态又是什么呢，前面说了有系统态和用户态，存放在哪？就存放在程序状态寄存器（PSW）中。



问题：中断到底什么时候发生？

CPU 执行指令的过程是取指执行，在每条指令执行前，都要进行检查（图中黄色圈），看是否有中断发生，怎么检查，就是在每条指令的最后时刻都扫描中断寄存器（图中红色圈），很简单是不是，如果有中断信号（CPU 其实很傻，只认信号），中断硬件机制就把程序状态寄存器（PSW）的相应位置 1，每个外设的中断都对应一个中断类型码（或叫中断号），如图所示：



如果真有中断发生，CPU 就转去执行中断处理程序，若无，就继续执行指令，这个过程循环往复。（参看 1.5 节，并查看相关资料）

问题：中断处理和子程序调用都需要压栈以保护现场，中断处理一定会保存而子程序调用不需要保存其内容的是（　）

A、程序计数器

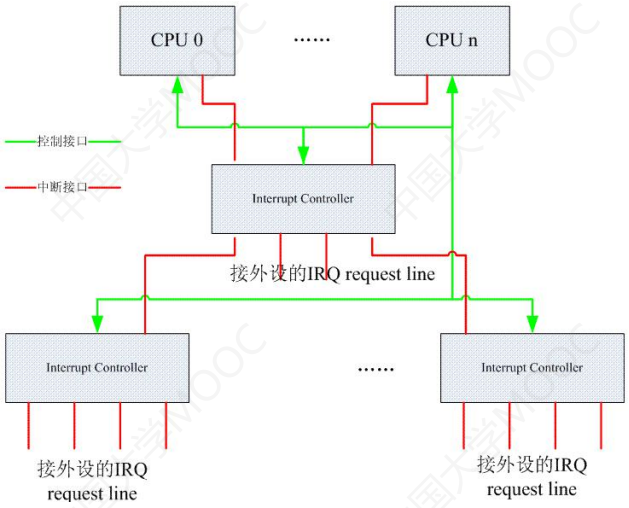
B、程序状态字寄存器

C、通用数据寄存器

D、通用地址寄存器

答：压栈保护现场这件事就像出了车祸要保存现场一样，中断处理程序与一般程序有什么相同呢，都是 CPU 本来好好的执行指令，然后有事情来打扰 CPU，就需要把 CPU 的现场保存下来，以便回来的时候知道当时离开的时候做什么。程序计数器（PC）存放程序第一条指令的执行地址，不管是中断还是子程序调用都需要保存，而通用数据寄存器和通用地址寄存器是压栈时都需要保存的，只有程序状态字寄存器（PSW）才是中断处理需要保存的内容。（参看 1.5 节）

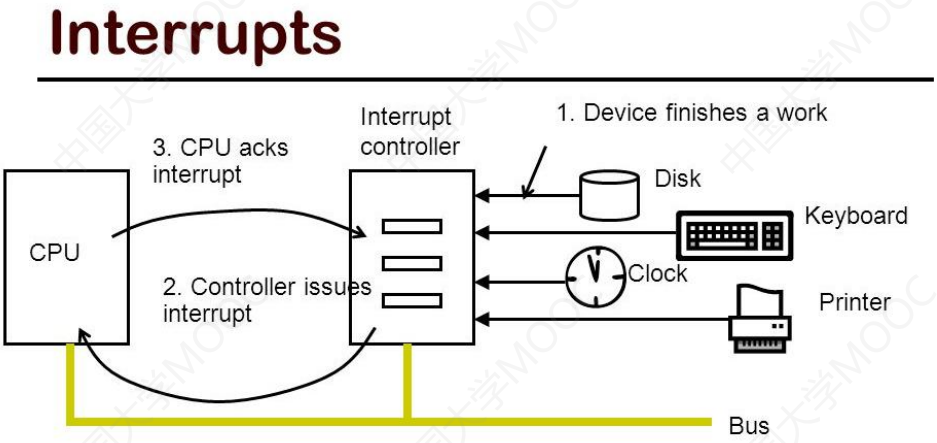
问题：中断模型是一种 C/S 结构吗？



C/S 模型的本质就是请求/服务模型，我们看中断模型，中断控制器相当于中介，把外设与 CPU 链接在一起，只要外设发出中断请求，CPU 就要去响应，也即是去服务，所谓服务，就是执行中断处理程序，是不是一种 C/S 模型？（参看中断概述）

问题：为什么要引入中断？

当你通过键盘敲了个字符，尽管你以为敲的飞速，但在 CPU 看来，你慢如蜗牛，我 CPU 总不能死等你键盘吧，CPU 在你敲键盘时，趁机运行微信了，这样看来，似乎 CPU 与键盘并行操作，而实际上，只是当你把字符敲完后，赶紧向 CPU 发了个中断请求，CPU 才来处理你的请求，看见了吧，中断的引入使得外设可以与 CPU 看起来并行工作，效率，又是效率，这不是操作系统始终追求的目标么！（参看中断概述）



问题：系统调用的实现从用户态切换到内核态，执行完系统调用程序后又从内核态切换回用户态，对系统调用进行优化是因为这种来回切换的代价很大吗？（参看 1.5 节）

系统调用与中断在处理机制上非常类似，但有所不同，中断是程序在执行的过程中，随时被打断，正在执行的进程可能在用户态也可能在内核态，但系统调用不同，只有用户态程序才能调用系统调用，比如 dos 中的 int 21H（参看 Hello world 视频），Unix/Linux 中的 read（）函数，调用了系统调用后，执行权就从用户态切换到内核态，切换到底有什么代价，想想美国总统换届就知道了，不管是中断、异常和系统调用、切换都需要保留现场，返回后要恢复现场，能否对陷入内核，从内核返回的速度稍微提升一点，因为一个程序执行会调用很多系统调用，这一点可以通过 Linux 下的 trace 命令追踪，比如，trace ls（追踪 ls 命令），可以看到调用了多少系统调用，因此说， 底层稍微一点的优化，对整个系统性能的影响会非常巨大。

问题：从哪个角度会使用到操作系统的系统调用（ ）

A、使用者

B、普通开发者

C、操作系统设计者

D、链接程序设计者

从这个图可以看出，一般使用者通过图形接口和命令终端使用计算机，开发者通过程序接口使用操作系统提供的系统调用，而链接程序设计者是设计链接程序把编译后的二进制代码链接起来，操作系统设计者是对操作系统的进程管理，内存管理，设备管理，文件管理进行设计，并给用户提供系统调用接口。（参看 1.2 节）

问题：系统调用是（　　）。

A、一条机器指令

B、中断子程序

C、用户子程序

D、提供编程人员的接口

给大家看一下 Linux 中系统调用 getpid()在内核中的实现：

```
long sys_getpid(void)
{
    return current->pid;
}
```

眼见为实，我们看到，获得进程的 pid 在内核的实现就是一个函数，给用户提供的接口为 getpid()，答案是什么，大家可以自己判断了。

问题：CPU 在执行过程中出错了叫什么？

CPU 是老大，它会不会出错，如果在程序执行的过程中它出错了怎么办？我们必须假定任何事情都会出错，而且 CPU 出错的话事态比较严重，因此给起了个名字就叫异常。

问题：能不能在哪看看系统中到底发生了哪些中断？

可以，命令：　cd /proc

　　cat interrupts

这里会列出很多信息，你可以自己了解这些信息到底代表了什么。

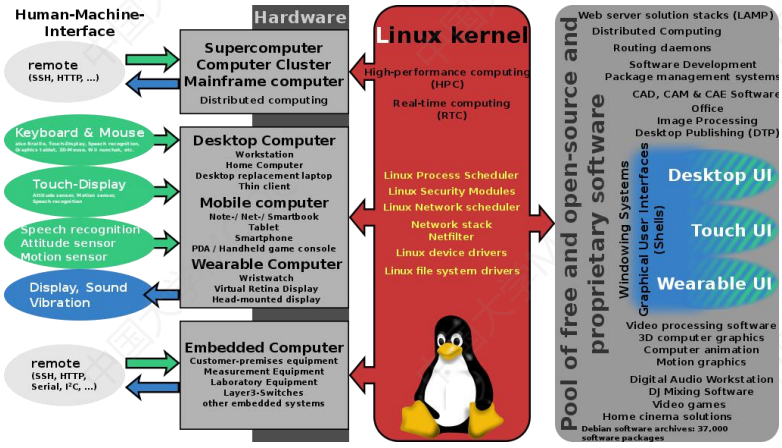
1.6 操作系统的运行环境和运行机制

查看视频：1.1.6 操作系统的运行环境和运行机制

2. 学堂在线：动手实践-编写 Linux 内核模块

问题：为什么要学 Linux 内核？

因为这里是全世界大牛的作品集合，有多少人参与？2 万左右的顶级大神，100 多个国家，google，微软，红帽子，Intel，华为等全球大厂都在这里争先恐后提交代码（有排名），以排名先后彰显自己的技术实力？应用在哪，看这张图就知道，无处不在，嵌入式，可穿戴计算机，移动计算机，桌面，到计算机集群和超级计算机。



问题：Linux 内核作为一个宏内核，能否方便的写代码并合并到内核中？

你问的问题就是 Linus 早都想到的，玩过积木吧，Linux 内核可加载模块就可以做到这点，通过内核模块，你就可以进入内核态写程序了，这可是从来没有干过的事，尽管你原来写了很多程序，也都是在用户态敲敲代码。进入内核写代码，与大牛思维大碰撞，你不愿意？

问题：Linux 内核模块编程要注意哪些事项？

首先说权限吧，你有 root 权限才能把内核模块插入到内核，其次，你在用户态用的那些函数库不能使用了，为啥？因为你现在打怪升级了，有特权了，不能再还想着原来哪些碎碎念念的事，现在要与大牛为伍了，先阅读内核的编码风格吧，不管能否出高质量的代码，但代码长的样子一定要高大上，与国际接轨，最后，动手实践吧，参看学堂在线《Linux 内核分析与应用》1.5 节动手实践-写一个 Linux 内核模块，顺便告诉你，你可以看看内核中的求最大数怎么写，就明白了什么叫高大上了。

Linux内核模块与C应用的对比

	C语言应用程序	内核模块程序
使用函数	Libc库	内核函数
运行空间	用户空间	内核空间
运行权限	普通用户	超级用户
入口函数	main()	module_init ()
出口函数	exit()	module_cleanup()
编译	gcc –c	make
连接	gcc	insmod
运行	直接运行	insmod
调试	gdb	kdbug, kdb,kgdb等