本部分主要一些操作系统相关的概念性的知识点，在后面的章节中，会提及这些知识点或对其进行详述。

**操作系统概念**

√ 界面或shell

基于图标的图形用户界面和基于文本的shell，不是操作系统的一部分。

√ 怎么看待操作系统:

从程序的角度看：操作系统的一个功能是为用户程序提供硬件的抽象，用户不需要了解各种硬件设备的运作方式及其驱动程序接口，通过调用操作系统提供的标准方法，就可以使用各种硬件设备。

从硬件的角度看：现代计算机包括处理器、存储器、时钟、磁盘、鼠标、网络接口、打印机及许多其他设备，操作系统需要管理这些资源，使其合理的分配给给不同的程序。

**计算机硬件介绍**

**● CPU：**

√ CPU工作方式

1 最简单的CPU工作方式是：

从内存中取一条指令 --> 对指令解码以确定其操作指令和操作数

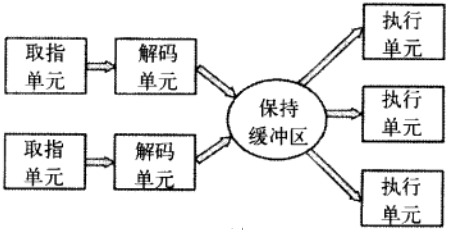
--> 执行该指令 --> 从内存中取下一条指令

2 采用流水线机制的CPU：

CPU具有分开的取指（取指令）单元、解码单元和执行单元，在同一时间，当解码单元在解码第n条指令时，执行单元正在处理第n-1条（前一条）指令，而取指单元已经开始获取第n+1条（后一条）指令。

在多数的流水线设计中，一旦一条指令进入流水线，他就必须执行完，即使前一条取出的是跳转指令，这样的设计客观操作造成来操作系统的复杂性。

3 超标量CPU：



这种设计的特点是，多天指令同时被取出、解码，然后将解码后的指令放在一个缓冲区中，然后各个执行单元从缓冲区中取出指令执行，CPU负责保证执行结果与顺序执行指令的结果一致。因为大部分的指令都是有因果关系的，所以这种超标量设计提升的效率是有限的。

值得一提的是，超标量设置中暗含来流水线设计，当某个解码单元在解码使，其对应的取指单元已经再取下一条指令了。

注意，超标量设计和多核设计是不同的，超标量使用的是一个程序计数器，一套控制逻辑，而多核CPU会有多个程序计数器，多个控制逻辑。

√ 寄存器

CPU中的寄存器可大体分为两类：通用寄存器和专用寄存器

通用寄存器：通用寄存器用来保存CPU运算的临时数据和运算结果：一部分CPU指令用于在内存和通用寄存器之间传递指令或数据，另一部分CPU指令可以对寄存器、内存中的数据运算产生运算结果。

专用寄存器：有程序计数器（PC）,程序状态奇（PSW），堆栈指针等。堆栈指针的个数和程序计数器的个数是一致的，它指向的是当前PC指向代码所使用的堆栈，堆栈用于存放函数参数、局部变量等。程序状态字中存放比较运算的结果、CPU模式（标明用户态、内核态）及各种控制位。

√ 内核态和用户态：

用户态和内核态是通过设置cpu的状态字控制的

内核态下，cpu可以执行所有的指令；用户态下，CPU只能执行部分指令，如与硬件设备直接通信的I/O指令，在用户态下就是不可用的，而且，在用户态下，内存的访问也是局限于用户空间。

一般来说，操作系统提供硬件抽象，所以他要与硬件直接打交道，故而操作系统通常是工作在内核态下的，在内核态可以通过设置CPU状态字切换到用户态；而用户程序则只能工作在用户态下，因为用户程序没有能直接进入内核态的方法，从用户态进入内核态的途径是中断、异常或陷入。

在核心态下，处理机不能被抢占，在用户态下，处理机和可以被抢占（被系统调动程序调度）。

有必要指出，计算机使用陷阱，而不是一条指令来执行系统调用。

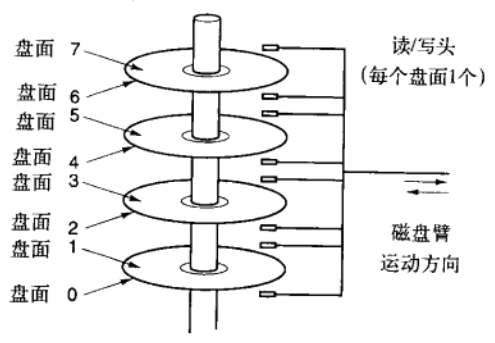
√ 多核CPU：

**● 存储器**

√ 高速缓存

现阶段的高速缓存通常都集成在CPU中，并设有多级缓存（目前一般为3级）：一级缓存（L1）又分为一级数据缓存和一级指令缓存，分别用于存放数据及执行数据的指令解码，两者可同时被CPU访问，其典型大小是16K，I7-8700K的L1为32KB+32KB；二级缓存的主要作用为，当CPU在L1中没读取到所需要的数据时再把数据展示给CPU筛选（CPU未命中L1的情况下继续在L2寻求命中），在多核CPU对二级缓存的处理上，Intel和AMD采用来不同的方案，因特尔采用的多核共享二级缓存的方案，而AMD则设计为每个核各有一个独立的二级缓存；同理，L3即为L2与主内存之间的缓冲器，主要体现在提升处理器大数据处理方面的性能，对游戏表现方面有较大的帮助。

√ 磁盘



通过图可以看出来，每个盘面都对应有一个读写头，但共用一个机械臂。

每个盘面从外到内分为一圈一圈的磁道，一个盘面的磁道数通常有上万条；磁道又分为一段段的扇区，外层磁道比内层磁道具有更多的扇区数，平均起来，每个磁道有几百个扇区；各个盘面第n环的磁道组合成第n柱面，之所以引入柱面这个单位，是因为选择一个读写头要比移动定位机械臂的速度快很多，所以磁盘在存放数据时，并不是用完了一个盘面再用下一个，而是用完一个盘面的第n磁道后，再用下一盘面的第n磁道，直到把最后一个盘面的第n磁道用完后，才移动磁盘臂，向内移动一个磁道，并从盘面0的磁道开始读写，所以，可以认为磁盘读写完一个柱面后，再开始读写下一个柱面。机械臂再柱面间移动是毫秒级的。

√ 堆和栈的区别

堆和栈都是存在内存中的，所以两者在存储速度上是一致的，不同的只是管理方式。

栈的申请和释放是由系统维护的，相如函数参数、局部变量等，都需要系统维护其空间的申请与释放，所以他们是存放在栈区的，另一方面，既然把局部变量、函数参数托付给系统来维护，那一个前提就是，系统需要事先知道（而不是等到真正执行到某条指令时才能确定）需要为存入栈区的变量分配多大的空间，所以要申请存放于栈中的数组局部变量，需要明确指明数组的长度。栈中的数据，一旦超出其作用范围，如函数局部变量在函数退出后，几乎立即会将该数据从栈空间中清除出去。

● **IO设备**

IO设备一般包括两个部分：设备控制器和设备本身。

√ 控制器：

控制器是一块或一组芯片，他从系统接收命令，然后控制设备读写数据，并对数据完成一定程度的处理。控制器完成对设备的复杂操作，为系统提供相对简单使用方式。例如磁盘控制器，他可以对外提供一个获取指定扇区数据的接口，要完成这一功能，控制器要完成把逻辑扇区映射为物理扇区、坏扇区重定位、移动磁盘臂、选择盘面磁头、计算校验和、把数据放到临时存储区等一系列操作，因此，在控制器中经常安装一个小的嵌入式计算机，该嵌入式计算机运行专门编号的程序来完成完成这些工作。

√ 设备：

设备本身也按标准规范提供来一个相对简单的控制接口，之所以为设备接口定义规范，是为了每一类控制器能通用的适配某一个设备。

√ 驱动程序：

设备驱动程序负责与控制器通信，每个控制器厂家负责为其支持的操作系统提供相应的驱动。设备驱动程序因为直接与IO设备打交道，所以要工作在内核态。

在Linux中，设备驱动安装时，会把一组设备驱动函数（open,close,read,write,ioctl等，在[file\_operation](https://blog.csdn.net/ldan508/article/details/50525955)结构体中定义）通过模块初始化函数注册给操作系统，模块退出函数则用于取消内核注册，释放资源。

那么什么时候模块初始化函数才获得运行呢？动态加载时，即运行insmode时。静态加载时模块编译进内核系统初始化时会自动调用这个模块初始化函数。

√ 中断：

中断源是硬件产生的中断，中断是由内核响应的，需要给这个中断注册一个中断处理函数。这可以通过调用linux内核提供特定的系统功能函数进行。 一般注册中断处理函数，可以在模块初始化函数里实现，或者在设备驱动接口程序open中实现。

√ 中断号：

[外部设备](https://baike.baidu.com/item/%E5%A4%96%E9%83%A8%E8%AE%BE%E5%A4%87" \t "https://baike.baidu.com/item/_blank)进行I/O操作时，会产生[中断请求](https://baike.baidu.com/item/%E4%B8%AD%E6%96%AD%E8%AF%B7%E6%B1%82" \t "https://baike.baidu.com/item/_blank)信号。这个信号中会有特定的标志，使计算机能够判断是哪个设备提出[中断请求](https://baike.baidu.com/item/%E4%B8%AD%E6%96%AD%E8%AF%B7%E6%B1%82" \t "https://baike.baidu.com/item/_blank)，这个信号就叫做中断号。

中断号是系统分配给每个中断源的代号，以便识别和处理。

中断号在中断处理过程中起到很重要的作用，在采用向量中断方式的中断系统中，CPU必须通过它才可以找到中断服务程序的入口地址，实现程序的转移。为了在中断向量表中查找中断服务程序的入口地址，可由中断号(n)×4得到一个指针，指向中断向量(即中断服务程序的入口地址)存放在中断向量表的位置，从中取出这个地址(CS：IP)，装入代码段寄存器CS和指令指针寄存器IP，即转移到了中断服务程序。

当一个设备需要CPU来处理它的数据时，可以向CPU发出中断请求信号，让CPU暂停正在执行的工作，转而处理该设备的操作请求，处理完毕后，再返回执行原来的工作。

当一个设备向CPU发出中断请求时，是通过IRQ值来告知CPU到底是哪一个外围设备需要服务，因此每个设备都会占用一个IRQ (Interrupt Request) 值。

在每台电脑的系统中，是由一个中断控制器8259或是8259A的芯片(现在此芯片大都集成到其他的芯片内)来控制系统中每个硬件的中断控制。目前共有16组IRQ，去掉其中用做桥接的一组IRQ，实际上只有15组IRQ可供硬件调用，而这些IRQ都有自己默认的配置。

在使用IRQ值时，需把握一条原则，即一个IRQ值只能给一个设备使用，例如：通常鼠标使用IRQ4(COMl)，此时IRQ4就不能再给其他硬件设备使用了。不同的操作系统版本对于IRQ的设置也不尽相同，所以在安装新硬件的时候，系统往往并不能完全自动检测正确的IRQ来分配给所调用的硬件，这就会造成此硬件设备或是原来的旧硬件出现不能正常工作的现象。其实这是系统自动将该硬件的IRQ分配给了其他与此IRQ相同的硬件，从而发生冲突使硬件不能正常工作。一般如果遇到这种情况，只要将新旧两个硬件的IRQ配置手动调开就可以解决了。