操作系统逻辑地址、线性地址和物理地址

转载 GeekWill 2016-09-06 14:45:22

文章标签： windows 内存 操作系统

本文转自论文点击打开链接，主要讲述了操作系统中人们应该熟悉但是容易混淆的逻辑地址、线性地址和物理地址：

1、逻辑地址：

逻辑地址（Logical Address）：是指由程序产生的与段相关的偏移地址部分。例如，你在进行C语言指针编程中，可以读取指针变量本身值(&操作)，实际上这个值就是逻辑地址，它是相对于你当前进程数据段的地址，不和绝对物理地址相干。只有在Intel实模式下，逻辑地址才和物理地址相等（因为实模式没有分段或分页机制,Cpu不进行自动地址转换）；逻辑也就是在Intel 保护模式下程序执行代码段限长内的偏移地址（假定代码段、数据段如果完全一样）。程序员仅需与逻辑地址打交道，而分段和分页机制对您来说是完全透明的，仅由系统编程人员涉及。应用程序员虽然自己可以直接操作内存，那也只能在操作系统给你分配的内存段操作。

如果是程序员，那么逻辑地址对你来说应该是轻而易举就可以理解的。我们在写C代码的时候经常说我们定义的结构体首地址的偏移量，函数的入口偏移量，数组首地址等等。当我们在考究这些概念的时候，其实是相对于你这个程序而言的。并不是对于整个操作系统而言的。也就是说，逻辑地址是相对于你所编译运行的具体的程序（或者叫进程吧，事实上在运行时就是当作一个进程来执行的）而言。你的编译好的程序的入口地址可以看作是首地址，而逻辑地址我们通常可以认为是在这个程序中，编译器为我们分配好的相对于这个首地址的偏移，或者说以这个首地址为起点的一个相对的地址值。

当我们双击一个可执行程序时，就是给操作系统提供了这个程序运行的入口地址。之后shell把可执行文件的地址传入内核。进入内核后，会fork一个新的进程出来，新的进程首先分配相应的内存区域。这里会碰到一个著名的概念叫做Copy On Write，即写时复制技术。这里不详细讲述，总之新的进程在fork出来之后，新的进程也就获得了整个的PCB结构，继而会调用exec函数转而去将磁盘中的代码加载到内存区域中。这时候，进程的PCB就被加入到可执行进程的队列中，当C

我们大可以把程序运行的入口地址理解为逻辑地址的起始地址，也就是说，一个程序的开始的地址。以及以后用到的程序的相关数据或者代码相对于这个起始地址的位置（这是由编译器事先安排好的），就构成了我们所说的逻辑地址。逻辑地址就是相对于一个具体的程序（事实上是一个进程，即程序真正被运行时的相对地址）而言的。尽管我们这样理解可能有一些细节上的偏差，但是比起网上一些含糊其辞，让人不知所云的描述要好得多，实用得多，等到自己对这个地址有更加深刻的理解的时候，再对上面的理解进行一些补充或者纠正。

总之一句话，逻辑地址是相对于应用程序而言的。

逻辑地址产生的历史背景：

追根求源，Intel的8位机8080CPU，数据总线（DB）为8位，地址总线（AB）为16位。那么这个16位地址信息也是要通过8位数据总线来传送，也是要在数据通道中的暂存器，以及在CPU中的寄存器和内存中存放的，但由于AB正好是DB的整数倍，故不会产生矛盾！

但当上升到16位机后，Intel8086/8088CPU的设计由于当年IC集成技术和外封装及引脚技术的限制，不能超过40个引脚。但又感觉到8位机原来的地址寻址能力2^16＝64KB太少了，但直接增加到16的整数倍即令AB＝32位又是达不到的。故而只能把AB暂时增加4条成为20条。则

2^20＝1MB的寻址能力已经增加了16倍。但此举却造成了AB的20位和DB的16位之间的矛盾，20位地址信息既无法在DB上传送，又无法在16位的CPU寄存器和内存单元中存放。于是应运而生就产生了CPU段结构的原理。

2、线性地址：

线性地址(linear address)或也叫虚拟地址(virtual address)： 线性地址（Linear Address）是逻辑地址到物理地址变换之间的中间层。程序代码会产生逻辑地址，或者说是段中的偏移地址，加上相应段的基地址就生成了一个线性地址。如果启用了分页机制，那么线性地址可以再经变换以产生一个物理地址。若没有启用分页机制，那么线性地址直接就是物理地址。Intel 80386的线性地址空间容量为4G（2的32次方即32根地址总线寻址）。

我们知道每台计算机有一个CPU（我们从单CPU来说吧。多CPU的情况应该是雷同的），最终所有的指令操作或者数据等等的运算都得由这个CPU来进行，而与CPU相关的寄存器就是暂存一些相关信息的存储记忆设备。因此，从CPU的角度出发的话，我们可以将计算机的相关设备或者部件简单分为两类：一是数据或指令存储记忆设备（如寄存器，内存等等），一种是数据或指令通路（如地址线，数据线等等）。线性地址的本质就是“CPU所看到的地址”（进程虚拟地址空间）。

如果我们追根溯源，就会发现线性地址的就是伴随着Intel的X86体系结构的发展而产生的。当32位CPU出现的时候，它的可寻址范围达到4GB，而相对于内存大小来说，这是一个相当巨大的数字，我们也一般不会用到这么大的内存。那么这个时候CPU可见的4GB空间和内
就是用

知道在多进程操作系统中，每个进程拥有独立的地址空间，拥有独立的资源。但对于某一个特定的时刻，只有一个进程运行于CPU之上。此时，CPU看到的就是这个进程所占用的4GB空间，就是这个线性地址。而CPU所做的操作，也是针对这个线性空间而言的。之所以叫线性空间，大概是因为人们觉得这样一个连续的空间排列成一线更加容易理解吧。其实就是CPU的可寻址范围。

3、物理地址：

物理地址（Physical Address）：是指出现在CPU外部地址总线上的寻址物理内存的地址信号，是地址变换的最终结果地址。如果启用了分页机制，那么线性地址会使用页目录和页表中的项变换成物理地址。如果没有启用分页机制，那么线性地址就直接成为物理地址了。

逻辑地址与物理地址的“差距”是0xC0000000，是由于虚拟地址->线性地址->物理地址映射正好差这个值。这个值是由操作系统指定的。

虚拟地址到物理地址的转化方法是与体系结构相关的。一般来说有分段、分页两种方式。以现在的x86 cpu为例，分段分页都是支持的。

Memory Mangement Unit负责从虚拟地址到物理地址的转化。逻辑地址是段标识+段内偏移量的形式，MMU通过查询段表，可以把逻辑地址转化为线性地址。如果cpu没有开启分页功能，那么线性地址就是物理地址；如果cpu开启了分页功能，MMU还需要查询页表来将线性地址转化为物理地址：

逻辑地址 ----（段表）---> 线性地址 —（页表）—> 物理地址

不同的逻辑地址可以映射到同一个线性地址上；不同的线性地址也可以映射到同一个物理地址上；所以是多对一的关系。另外，同一个线性地址，在发生换页以后，也可能被重新装载到另外一个物理地址上。所以这种多对一的映射关系也会随时间发生变化。