为何要把物理内存地址映射到内核空间？

发表于2010/6/4 11:18:00  707人阅读

分类： Windows linux

提出这个问题是因为32位的x86如果没有特殊的支持和机制，是无法访问896M以上的物理内存的，这让我很奇怪。虽然内核所分配到的空间是[3G,4G-1)，但是这并不代表内核本身的寻址能力是1G呀。  
  
后来看到一篇文章说，linux会在初始化的时候把物理内存映射到内核空间。这也让我很奇怪。linux本身维护了一个page的数组来作为物理内存的“仓库”，这个数组的下标实际上就是物理内存的页面号，linux会在初始化的时候初始化这个数组。我的理解是，后期的虚存到物理内存的映射实际上都落实到了对这个page数组的操作，这就够了。为什么还要把物理内存地址映射到内核空间？  
  
并且，为什么仅仅是896M，剩下的128M内核地址空间去哪儿了？难道剩下的这128M就是内核本身么？如果是的话，那么内核中的各种系统调用、内核数据结构等就都要挤在这个空间里，这样要对这相对所剩无几的空间进行良好的规划，而不是像普通程序那样，一开始就是8开头的虚存空间地址；如果不是的话，那这128M方的又是什么东西，内核自己又躲到哪儿去了？  
  
  
  
  
**linux这样做，只是为了简化内核中虚拟地址和物理地址之间相互转化的工作，\_\_va()、\_\_pa()轻松的就变换过来了，并不是说物理内存映射到内核空间就全部被内核占了。对于内核中非动态数据使用的物理内存是不可以变了，但是动态的部分，是可以释放掉的，而释放掉后这一部分对应的物理内存又可以影射到用户空间**。  
  
至于为什么不完全映射1g的空间，我想是因为还要留一片内核的虚拟地址给ioram用，比如说pci的ioram，通过ioremap映射总得留下点虚拟地址把，而ioram的虚拟地址又必需在内核空间中，也就是3-4g。   
  
不完全映射1G，是因为要保留出一段空间来供动态映射所使用，比如当内存大于1G是，除了前面的890M可以直接映射，后面的空间都需要临时映射一下，用完解除映射(某些平台下实际解除是个空操作)。  
  
对于刚好1G内存的时候，动态映射其实是没有必要的。www.kerneltrap.org上曾经有文章讨论过一个patch，可以去掉动态映射，但只能用于少于1G的情况，好像现在kernel的配置已经有这个选项了。  
  
内核的地址是从物理内存的0地址开始的，而0xc0000000就是这个偏移量  
  
"linux这样做，只是为了简化内核中虚拟地址和物理地址之间相互转化的工作，\_\_va()， \_\_pa()轻松的就变换过来了。 "  
  
  
物理地址 <--> 虚拟地址  
#define \_\_pa(x) ((unsigned long)(x) & 0x3fffffff)  
#define \_\_va(x) ((void \*)((unsigned long)(x) | 0xc0000000))