# [x86分页机制](http://blog.csdn.net/longintchar/article/details/52072253)

1. 为什么会有分页机制？

当同时运行的任务很多时，内存可能就不够用。每个段描述符都有A位，每当访问一个段时，处理器都会将其置位。操作系统可以定时对A位清零，借此机会可以统计段的访问频率。当内存不够用的时候，操作系统可以将访问较少的段换出到磁盘上，以腾出空间给马上要运行的段使用。当某个段被换出到磁盘时，操作系统应该将这个段的描述符的P位清零。过上一段时间，当再次访问这个段时，因为它的描述符的P位是0，处理器就会引发段不存在异常（中断号11）。这类中断通常是由操作系统处理的，它会用同样的方法腾出空间，然后把这个段从磁盘调入内存。当这类中断返回时，处理器会再次执行引发异常的那条指令，这时候段已经在内存中（P=1），于是程序又可以继续执行了。

由此可见，即使没有分页机制，利用“分段”也可以实现“虚拟内存”。

但是，因为段的长度不固定，在段的换入换出时会产生外部碎片，这样就浪费了很多内存。为了解决这个问题，从80386处理器开始，引入了分页机制。

分页机制简单来说，是用长度固定的页来代替长度不定的段，以解决因段的长度不同带来的内存空间管理变得复杂的问题。

尽管操作系统也可以利用纯软件来实施固定长度的内存分配，但是太过于复杂。由处理器固件来做这件事情，可以省去很多麻烦，速度也可以提高。

2. 从虚拟地址到物理地址

分页机制是80x86内存管理机制的第二部分。它在分段机制的基础上完成虚拟（逻辑）地址到物理地址转换的过程。分段机制把逻辑地址转换成线性地址，而分页则把线性地址转换成物理地址。 

分页机制会把线性地址空间（段已映射到其中）划分成页面，然后这些线性地址空间的页面被映射到物理地址空间的页面上。

80x86使用4K（2的12次方）字节固定大小的页面。每个页面均是4KB，且对齐于4K地址边界处（地址的低12位全是0）。

3. 简单的分页模型

4GB（2的32次方）的线性地址空间可以划分为1048576（2的20次方，即1M）个页面。为了根据线性地址找到对应的物理地址，操作系统必须维护一张表（如下图所示）。 

这个表暂且叫做“页映射表”，它一共有1048576个表项，每个表项占4个字节，其内容是某个页的起始物理地址（共32比特，低12位全为0）.页映射表是这样使用的：因为页的尺寸是4KB，所以线性地址的低12位可以作为页内偏移，高20位可以用来索引一个表项，找到了这个表项，就找到了对应的物理页。

4. 页表和页目录

4.1 层次化的分页结构

上文提到的页映射表，一共有1048576（=1M）个表项，每个表项占4个字节，所以表的大小是4MB，在当时看来要占用相当一部分内存。考虑到在实践中，没有哪个任务会真的用到所有表项，充其量只是很小一部分，所以内存中放一个4MB的表格确实很浪费。也许你会建议，能不能先划出一小片内存，只存表格用到的部分，然后根据需要动态扩展。的确，这个方法可行。但是因为特殊原因（后面我们会学习到，整个映射表的前一半对应全局地址空间，后一半对应局部地址空间），这张表从一开始就必须完全定义（留够空间），所以不可避免地要占用4MB的内存空间。为了解决这个问题，同时又不会浪费宝贵的内存空间，处理器设计了层次化的分页结构。

4.2 页表

4GB（2的32次方）的线性地址空间可以划分为1048576（2的20次方，即1M，也可以看成是1024\*1024）个页面，所以，可以随机地抽取这些页面，每1024个页面是一组,可以分成1024组。对于每组中1024个页面的物理地址，按某种顺序排列可以构成一张表（每个表项都是一个页面的物理地址），这个表就是页表。页表的大小是1024\*4B=4KB，刚好是一个物理页的大小。

4.3 页目录

因为已经分成了1024组，每组都有一个页表（大小为4KB），所以这1024个页表又可以用一张表来指向，这就是页目录。类似于页表，页目录共有1024个表项（称作页目录项），每个页目录项的内容是某个页表的物理地址。页表的大小是1024\*4B=4KB，刚好是一个物理页的大小。

需要注意的是：   
1. 这样的层次化分页结构是每个任务都有的，或者说每个任务都有自己的页目录。在处理器内部，有一个控制寄存器叫CR3，存放着当前任务的页目录的物理地址，故CR3又叫做页目录基址寄存器（Page Directory Base Register，PDBR）.   
2. 每个任务都有自己的TSS，其中就包括了CR3寄存器域，存放着任务自己的页目录的物理地址。当任务切换时，CR3寄存器的内容也会被更新，更新为新任务的页目录的物理地址。   
3. 页目录和页表也是普通的页，混迹于全部的物理页中。它们和普通的页没有什么区别，无非就是功能不一样。当任务被操作系统撤销后，它们和任务所占用的普通的物理页一样会被回收。   
4. 页目录总是在物理内存中，页表可以在需要时再分配，这样就大大节省了物理内存。

5. 地址变换的具体过程

具体怎么变换，还是用书上的例子来说明吧。   
假设段部件输出的线性地址是0x00801050,如果没有开启分页，那么这个地址就是物理地址；但是现在开启了分页，所以要经过页部件的转换，才能得到物理地址。

处理器的页部件专门负责线性地址到物理地址的转换工作。它首先将32位的线性地址分成3段，分别是高10位，中间10位和低12位。高10位用来索引页目录，中间10位用来索引页表，低12位作为页内偏移。

1. 当前任务的页目录的物理地址在CR3寄存器中，假设是0x00005000;
2. 段部件输出的线性地址是0x00801050,按照高10位，中间10位和低12位分为三段，分别是0x002,0x001,0x050;
3. 0x002乘以4（因为每个表项占4个字节）得到0x008,作为偏移访问页目录，得到了0x08001000,这就是页表的物理地址。
4. 0x001乘以4（因为每个表项占4个字节）得到0x004,作为偏移访问页表，得到了0x0000c000,这就是我们要找的那个物理页的起始地址。
5. 0x050作为页内偏移，和物理页的起始地址0x0000c000相加，得到0x0000c050,这就是最终的物理地址。