<https://blog.csdn.net/weiyuefei/article/details/52089774>

1.什么是地址空间？

地址空间是一个新的内存抽象，就像进程的概念创造了一类抽象的CPU以运行程序一样。

地址空间是一个进程可用于寻址内存的一套地址集合。每个进程都有一个自己的地址空间，并且这个地址空间独立于其他进程的地址空间。

地址空间可以保证多个应用程序同时处于内存中并且不互相影响（保护和重定位）。

2. 什么是交换技术？

交换技术，即把一个进程完整调入内存，使该进程运行一段时间，然后把它存回磁盘。（用来处理内存超载）

空闲进程主要存储在磁盘上，所以当它们不运行时就不会占用内存。

3.空闲内存管理方式有哪些？

1）使用位图的存储管理：内存被划分成小到几个字或大到几千字节的分配单元，每个分配单元对应于位图中的一位，0表示空闲，1表示占用。

2）使用链表（双链表）的存储管理：维护一个记录已分配内存段和空闲内存段的链表。其中链表中的一个结点或者包含一个进程，或者是两个进程间的一个空闲区（即前一个结点是进程A，后一个结点是进程B，那么这个结点就是A和B之间的空闲区）。

4.什么是虚拟内存？

虚拟内存的基本思想是：每个程序拥有自己的地址空间，这个空间被分割成多个块，每一块称作为一页或页面。每一页有连续的地址范围。这些页被映射到物理内存，但并不是所有的页都必须在内存中才能运行程序。当程序引用一部分在物理内存中的地址空间时，由硬件立刻执行必要的映射。当程序引用到一部分不在物理内存中的地址空间时，由操作系统负责将缺失的部分装入物理内存并重新执行失败的指令。

把那些不常用的程序片段放入虚拟内存，当需要用到它的时候再load入主存（物理内存（代码的位置必须在物理内存中才能被运行））中。这个就是内存管理要做的事。

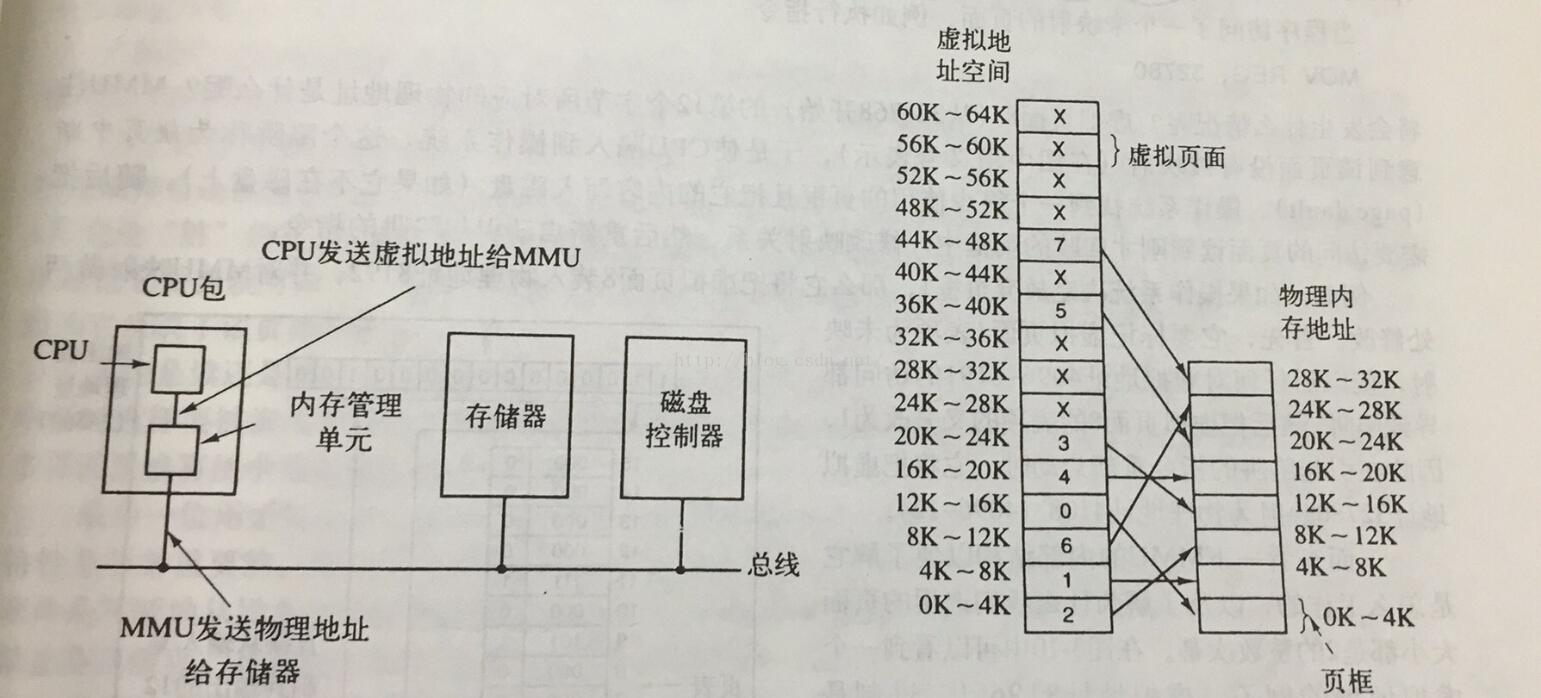
内存管理还有另外一件事需要做：计算程序片段在主存中的物理位置，以便CPU调度。

5.分页

由程序产生的地址称为虚拟地址，他们构成了一个虚拟地址空间。

虚拟地址空间按照固定大小划分成称为页面的若干单元。在物理内存中对应的单元称为页框，页面和页框的大小通常是一样的。

RAM和磁盘之间的交换总是以整个页面为单元进行的。



当程序访问一个已经映射到物理地址的页面时，可以根据映射直接访问对应的物理地址，当程序访问一个没有映射的页面时，MMU（内存管理单元）注意到该页面没有被映射，于是使CPU陷入到操作系统，这个陷阱称为缺页中断。操作系统找到一个很少使用的页框且把它的内容写入磁盘。随后把需要访问的页面读到刚才回收的页框中，修改映射关系，然后重新启动引起陷阱的指令。

6.虚拟地址到物理地址的映射

虚拟地址到物理地址的映射可以概括如下：虚拟地址被分成虚拟页号（高位）和偏移量（低位）两部分。虚拟页号可用作页表的索引，以找到该虚拟页面对应的页表项，由页表项找到页框号（如果有的话）。然后把页框号拼接到偏移量的最高端，以替换掉虚拟页号，形成送往内存的物理地址。

页表的目的是把虚拟页面映射为页框。

7.页面置换

当发生缺页中断时，操作系统必须在内存中选择一个页面将其换出内存，以便为即将调入的页面腾出空间。如果要换出的页面在内存驻留期间已经被修改过，就必须把它写回磁盘以更新该页面在磁盘的副本；如果该页面没有被修改过，那么它在磁盘上的副本已经是最新的，不需要回写。直接用调入的页面覆盖掉被淘汰的页面就可以了。

8.页面置换算法

1）最优页面置换算法（理想，不可能实现）：在缺页中断发生时，有些页面在内存中，其中有一个页面很快被访问，其他页面则可能要到10、100或1000条指令后才会被访问，每个页面都可以用在该页面首次被访问前所要执行的指令数作为标记。最有页面置换算法规定应该置换标记最大的页面。

2）最近未使用页面置换算法：随机地从类编号最小的非空类中挑选一个页面淘汰之。这个算法的隐含意思是，在最近一个时钟滴答中（典型的时间大约20ms）淘汰一个没有被访问的已修改页面要比淘汰一个被频繁使用的“干净”页面好。

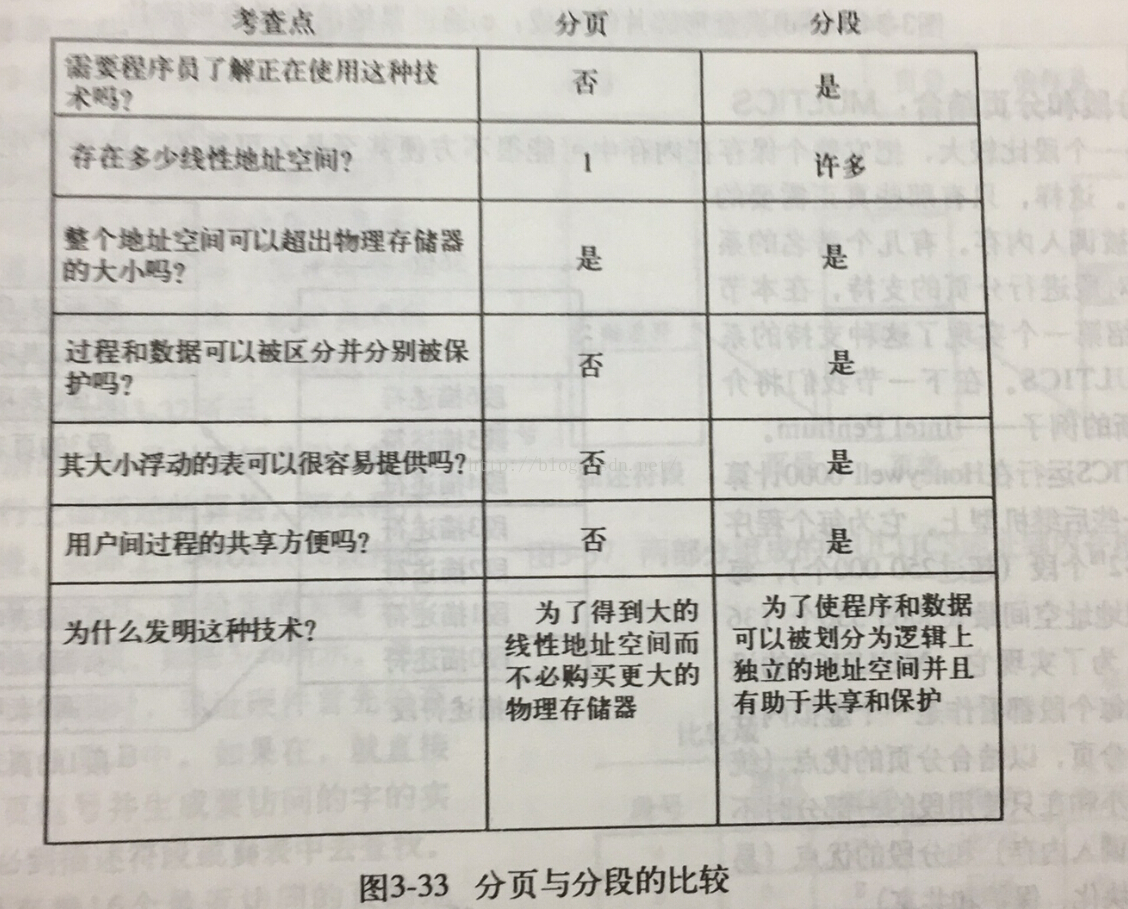
3）先进先出页面置换算法：由操作系统维护一个所有当前在内存中的页面的链表，最新进入的页面放在表尾，最久进入的页面放在表头。当放生缺页中断时，淘汰表头的页面并把新调入的页面加到表尾。

9.分段

段是互相独立的地址空间，每个段由一个从0到最大的线性地址序列构成，每个段的长度可以是0到某个允许的最大值之间的任何一个值。不同的段的长度可以不同，并且通常情况下也都不相同。段的长度在运行期间可以动态改变。比如堆栈段。

要在这种分段或二维的存储器中指示一个地址，程序必须提供两部分地址，一个段号和一个段内地址。

段不会同时包含多种不同类型的内容。



10.分段与分页结合

把主存分成若干页，每页分成若干段。

转自【http://blog.csdn.net/tianya\_team/article/details/50551208】