Atitit 存储管理 数据库文件 段页式管理 os操作系统

[1. 起源 内存管理》磁盘管理》数据库文件管理 1](#_Toc20758)

[2. 内存管理 1](#_Toc13118)

[2.1. 没有内存抽象的年代 2](#_Toc21866)

[2.2. 内存抽象 3](#_Toc28844)

[2.3. 虚拟内存(Virtual Memory)页(Page). 3](#_Toc18052)

[2.4. 内存管理中还有一种分段式管理，也就是一个进程可以拥有多个独立的逻辑地址， 4](#_Toc1877)

[3. 第3章 存储器管理 4](#_Toc4677)

[3.1. 3.1 存储管理的概念 4](#_Toc3073)

[3.2. 3.2 分区存储管理 c盘 d盘 4](#_Toc1813)

[3.3. 3.3 页式存储管理 4](#_Toc29999)

[3.4. 3.4 段式存储管理 6](#_Toc1100)

[3.5. 3.5 段页式存储管理 6](#_Toc9319)

# 起源 内存管理》磁盘管理》数据库文件管理

# 内存管理

统的角度来看，进程=程序+数据+PCB(进程控制块)。这个概念略微有点抽象，我通过一个类比来说吧：比如，你正在厨房做饭，你一边看着菜谱一边按照菜谱将原料做成菜，就在这时，你儿子进来告诉你他擦破了腿，此时你停下手中的工作，将菜谱反扣过来，然后找来急救书按照书中的内容给你儿子贴上创口贴，贴完后你继续回去打开菜谱，然后继续做饭。在这个过程中，你就好比CPU，菜谱就好比程序，而做菜的原料就好比数据。你按照程序指令加工数据，而急救工作好比一个更高优先级的进程，中断了你当前做饭的工作,然后你将菜谱反扣过来（保护现场）,转而去处理高优先级的进程，处理完毕后你继续从刚才的页读菜谱(恢复现场)，然后继续执行做菜这个进程。

    在简单介绍完进程的概念后，我们来转入内存。

## 没有内存抽象的年代

    在早些的操作系统中，并没有引入内存抽象的概念。程序直接访问和操作的都是物理内存。比如当执行如下指令时:

mov reg1,1000

    这条指令会毫无想象力的将物理地址1000中的内容赋值给寄存器。不难想象，这种内存操作方式使得操作系统中存在多进程变得完全不可能，比如MS-DOS，你必须执行完一条指令后才能接着执行下一条。如果是多进程的话，由于直接操作物理内存地址，当一个进程给内存地址1000赋值后，另一个进程也同样给内存地址赋值，那么第二个进程对内存的赋值会覆盖第一个进程所赋的值，这回造成两条进程同时崩溃。

    没有内存抽象对于内存的管理通常非常简单，除去操作系统所用的内存之外，全部给用户程序使用。或是在内存中多留一片区域给驱动程序使用，如图1所示。

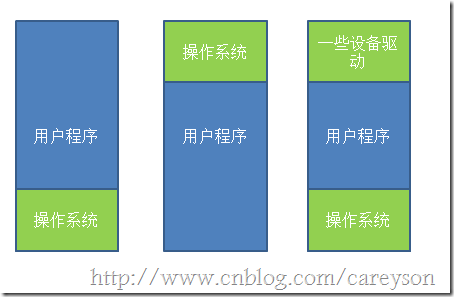
[](http://images.cnblogs.com/cnblogs_com/CareySon/201204/201204251637269578.png)

    图1.没有内存抽象时，对内存的使用

    第一种情况操作系统存于RAM中，放在内存的低地址，第二种情况操作系统存在于ROM中，存在内存的高地址，一般老式的手机操作系统是这么设计的。

    如果这种情况下，想要操作系统可以执行多进程的话，唯一的解决方案就是和硬盘搞交换，当一个进程执行到一定程度时，整个存入硬盘，转而执行其它进程，到需要执行这个进程时，再从硬盘中取回内存，只要同一时间内存中只有一个进程就行，这也就是所谓的交换（Swapping）技术。但这种技术由于还是直接操作物理内存，依然有可能引起进程的崩溃。

    所以，通常来说，这种内存操作往往只存在于一些洗衣机，微波炉的芯片中，因为不可能有第二个进程去征用内存。

## 内存抽象

    在现代的操作系统中，同一时间运行多个进程是再正常不过的了。为了解决直接操作内存带来的各种问题，引入的地址空间(Address Space),这允许每个进程拥有自己的地址。这还需要硬件上存在两个寄存器，基址寄存器(base register)和界址寄存器(limit register),第一个寄存器保存进程的开始地址，第二个寄存器保存上界，防止内存溢出。在内存抽象的情况下，当执行

mov reg1,20

    这时，实际操作的物理地址并不是20，而是根据基址和偏移量算出实际的物理地址进程操作，此时操作的实际地址可能是:

mov reg1,16245

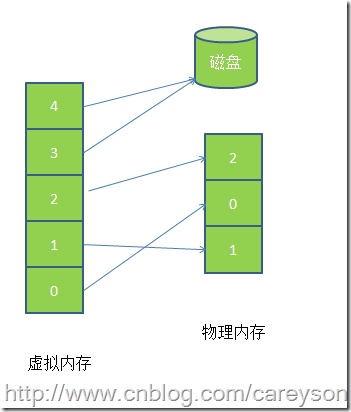
    在这种情况下，任何操作虚拟地址的操作都会被转换为操作物理地址。而每一个进程所拥有的内存地址是完全不同的，因此也使得多进程成为可能。

    但此时还有一个问题，通常来说，内存大小不可能容纳下所有并发执行的进程。因此，交换(Swapping)技术应运而生。这个交换和前面所讲的交换大同小异,只是现在讲的交换在多进程条件下。交换的基本思想是，将闲置的进程交换出内存，暂存在硬盘中，待执行时再交换回内存，比如下面一个例子，当程序一开始时，只有进程A，逐渐有了进程B和C，此时来了进程D，但内存中没有足够的空间给进程D，因此将进程B交换出内存，分给进程D。如图2所示。

## **虚拟内存(Virtual Memory)**页(Page).

    虚拟内存是现代操作系统普遍使用的一种技术。前面所讲的抽象满足了多进程的要求，但很多情况下，现有内存无法满足仅仅一个大进程的内存要求(比如很多游戏，都是10G+的级别)。在早期的操作系统曾使用覆盖(overlays)来解决这个问题，将一个程序分为多个块，基本思想是先将块0加入内存，块0执行完后，将块1加入内存。依次往复，这个解决方案最大的问题是需要程序员去程序进行分块，这是一个费时费力让人痛苦不堪的过程。后来这个解决方案的修正版就是虚拟内存。

    虚拟内存的基本思想是，每个进程有用独立的逻辑地址空间，内存被分为大小相等的多个块,称为页(Page).每个页都是一段连续的地址。对于进程来看,逻辑上貌似有很多内存空间，其中一部分对应物理内存上的一块(称为页框，通常页和页框大小相等)，还有一些没加载在内存中的对应在硬盘上，如图5所示。

[](http://images.cnblogs.com/cnblogs_com/CareySon/201204/201204251637532484.png)

## 内存管理中还有一种分段式管理，也就是一个进程可以拥有多个独立的逻辑地址，

以后有时间了再补上一篇。

# 第3章 存储器管理

## 3.1 存储管理的概念

## 3.2 分区存储管理 c盘 d盘

## 3.3 页式存储管理

内存分区存储管理的一个特点是连续性，每个程序都分有一片连续的内存区域。这种连续性导致碎片问题，包括

固定分区中的内碎片和可变分区中的外碎片。为了解决这些问题，人们又提出了“页式存储管理方案”。它的基本出发点

是打破存储分配的连续性，使一个程序的逻辑地址空间可以分布在若干个离散的内存块上，从而达到充分利用内存，提高

内存利用率的作用。

页式存储管理的基本思路是：一方面，把物理内存划分为许多个固定大小的内存块，称为物理页面，或页框；另一方面，把逻辑

地址空间也划成大小相同的块，称为逻辑页面，页面的大小要求是2^n.一般在512个字节到8KB之间。当一个用户程序被装入内存时，不是以

整个程序为单位，把它放在一整块连续的区域，而是以页面为单位来进行分配的。对于一个大小为N个页面的程序，需要有N个空闲的物理页面

（1）数据结构。

在页式存储管理中，最主要的数据结构有两个。

1、页表（page table）：页表给出了任务的逻辑页面号与内存中的物理页面号之间的对应关系。

2、物理页面表：用来描述内存空间中，各个物理页面的使用分配状况。在具体实现上，可以采用位示图或空闲页面链表等方法。

4、优缺点：

优点：1、没有外碎片。程序不必连续存放。便于管理。

缺点：程序必须全部装入内存，才可以运行。操作系统必须为每一个任务都维护一张页表，开销比较大，简单的页表结构已经

不能满足要求，必须设计出更复杂的结构。如多级页表结构、哈希页表结构、反置页表。

三、小结

    页式存储管理的优点是将作业的连续逻辑地址空间划分成页，可以分配到内存中不连续的块中，也就是分配到内存的不连续的主存区域中，并且能使作业正确的执行，这样进一步的提高了主存空间的利用率。缺点是当处理器处理一个作业时，必须访问两次主存，第一次是访问页表从而找到页号所对应的块号，然后换算出作业的绝对地址，第二次是处理器再按照换算出程序在主存中的绝地地址再进行运算操作

## 3.4 [段式存储管理](https://baike.baidu.com/item/%E6%AE%B5%E5%BC%8F%E5%AD%98%E5%82%A8%E7%AE%A1%E7%90%86" \t "https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F%E5%8E%9F%E7%90%86/_blank)

五、段式存储管理（ 领会 ）

　　1、段式存储中段的划分

　　段式管理 是根据人们对 程序 中需要 分段编制 的要求出发而提供的。它提供给用户编程时使用的逻辑地址由“段号”和“段内地址”两部分组成，其形式和页式管理相同。但是实际上是不同的：

　　页式存储管理提供连续逻辑地址由系统自动分页，段式存储管理中的作业分段是由用户决定的，每段独立编程，因此段间的逻辑地址是不连续的。

## 3.5 段页式存储管理

分段机制：

  在8086时期，寄存器是16位，无法存放20位的物理地址，为了解决这问题，人们提出了分段机制，分段机制就是将内存分段，每段大小64kb（正好由16位表示）,在段寄存器中放入段基址，然后+上段偏移量就成了物理地址。

分页机制：

Windows 2000 使用基于分页机制的虚拟内存。每个进程有4GB的虚拟地址空间。基于分页机制，这4GB地址空间的一些部分被映射了物理内存，一些部分映射硬盘上的交换文件，一些部分什么也没有映射。程序中使用的都是4GB地址空间中的虚拟地址。而访问物理内存，需要使用物理地址。

操作系统原理：页式存储管理 - CSDN博客.mhtml

操作系统原理---操作系统中的内存管理 - niupan369 - 博客园.mhtml

操作系统原理---操作系统中的内存管理 - niupan369 - 博客园.mhtml