## 传统存储管理方式的特征

各种内存管理策略都是为了同时将多个进程保存在内存中以便允许多道程序设计。它们都具有以下两个共同的特征：

#### 1) 一次性

**作业必须一次性全部装入内存后，方能开始运行。**这会导致两种情况发生：

* **当作业很大，不能全部被装入内存时，将使该作业无法运行；**
* **当大量作业要求运行时，由于内存不足以容纳所有作业，只能使少数作业先运行，导致多道程序度的下降。**

#### 2) 驻留性

**作业被装入内存后，就一直驻留在内存中，其任何部分都不会被换出，直至作业运行结束。**运行中的进程，会因等待I/O而被阻塞，可能处于长期等待状态。  
  
由以上分析可知，**许多在程序运行中不用或暂时不用的程序（数据）占据了大量的内存空间，而一些需要运行的作业又无法装入运行，显然浪费了宝贵的内存资源。**

## 局部性原理

要真正理解虚拟内存技术的思想，首先必须了解计算机中著名的局部性原理。著名的 Bill Joy (SUN公司CEO)说过：”在研究所的时候，我经常开玩笑地说高速缓存是计算机科学中唯一重要的思想。事实上，髙速缓存技术确实极大地影响了计算机系统的设计。“快表、页高速缓存以及虚拟内存技术从广义上讲，都是属于高速缓存技术。这个技术所依赖的原理就是局部性原理。局部性原理既适用于程序结构，也适用于数据结构（更远地讲，Dijkstra 著名的关于“goto语句有害”的论文也是出于对程序局部性原理的深刻认识和理解）。  
  
局部性原理表现在以下两个方面：

* 时间局部性：如果程序中的某条指令一旦执行，不久以后该指令可能再次执行；如果某数据被访问过，不久以后该数据可能再次被访问。产生时间局部性的典型原因，**是由于在程序中存在着大量的循环操作**。
* 空间局部性：一旦程序访问了某个存储单元，在不久之后，**其附近的存储单元也将被访问，即程序在一段时间内所访问的地址，可能集中在一定的范围之内**，这是因为指令通常是**顺序存放、顺序执行的**，数据也一般是以向量、数组、表等形式簇聚存储的。

时间局部性是通过**将近来使用的指令和数据保存到高速缓存存储器中**，并使用高速缓存的层次结构实现。空间局部性通常是**使用较大的高速缓存，**并将预取机制集成到高速缓存控制逻辑中实现。虚拟内存技术实际上就是建立了 **“内存一外存”的两级存储器的结构，利用局部性原理实现髙速缓存。**

## 虚拟存储器的定义和特征

基于局部性原理，**在程序装入时，可以将程序的一部分装入内存，**而将其余部分留在外存，就可以启动程序执行。在程序执行过程中，**当所访问的信息不在内存时，由操作系统将所需要的部分调入内存,然后继续执行程序。**另一方面，操作系统将内存**中暂时不使用的内容换出到外存上，从而腾出空间存放将要调入内存的信息。**这样，系统**好像为用户提供了一个比实际内存大得多的存储器，称为虚拟存储器。**  
之所以将其称为虚拟存储器，是**因为这种存储器实际上并不存在**，只是由于（对用户完全透明），给用户的感觉是好像存在一个比实际物理内存大得多的存储器。**虚拟存储器的大小由计算机的地址结构决定，**并非是内存和外存的简单相加。虚拟存储器有以下三个主要特征：

* 多次性，是指**无需在作业运行时一次性地全部装入内存**，而是允许被分成多次调入内存运行。
* 对换性，是指**无需在作业运行时一直常驻内存**，而是允许在作业的运行过程中，进行换进和换出。
* 虚拟性，是指**从逻辑上扩充内存的容量**，使用户所看到的内存容量，远大于实际的内存容量。

## 虚拟内存技术的实现

虚拟内存中，允许将一个作业分多次调入内存。釆用连续分配方式时，会使相当一部分内存空间都处于暂时或“永久”的空闲状态，造成内存资源的严重浪费，而且也无法从逻辑上扩大内存容量。因此，**虚拟内存的实现需要建立在离散分配的内存管理方式的基础上。虚拟内存的实现有以下三种方式：**

* **请求分页存储管理。**
* **请求分段存储管理。**
* **请求段页式存储管理。**

不管哪种方式，都需要有一定的硬件支持。一般需要的支持有以下几个方面：

* **一定容量的内存和外存。**
* **页表机制（或段表机制），作为主要的数据结构。**
* **中断机构，当用户程序要访问的部分尚未调入内存，则产生中断。**
* **地址变换机构，逻辑地址到物理地址的变换。**

# 3.6 请求分页管理方式实现虚拟内存

**请求分页系统**建立在基本分页系统基础之上，为了支持虚拟存储器功能而**增加了请求调页功能和页面置换功能**。**请求分页是目前最常用的一种实现虚拟存储器的方法。**  
在请求分页系统中，**只要求将当前需要的一部分页面装入内存，便可以启动作业运行。**在作业执行过程中，当所要访问的页面不在内存时，**再通过调页功能将其调入，同时还可以通过置换功能将暂时不用的页面换出到外存上，**以便腾出内存空间。  
  
为了实现请求分页，系统必须提供一定的硬件支持。除了需要一定容量的内存及外存的计算机系统，还需要有**页表机制、缺页中断机构和地址变换机构。**

## 页表机制

**请求分页系统的页表机制不同于基本分页系统，请求分页系统在一个作业运行之前不要求全部一次性调入内存，因此在作业的运行过程中，必然会出现要访问的页面不在内存的情况，**如何发现和处理这种情况是请求分页系统必须解决的两个基本问题。为此，**在请求页表项中增加了四个字段，**如图3-24所示。

http://c.biancheng.net/cpp/uploads/allimg/140701/1-140F10202531Y.png  
图3-24  请求分页系统中的页表项

增加的四个字段说明如下：

* 状态位P：用于指示该页是否已调入内存，供程序访问时参考。
* 访问字段A：用于记录本页在一段时间内被访问的次数，或记录本页最近己有多长时间未被访问，供置换算法换出页面时参考。
* 修改位M：标识该页在调入内存后是否被修改过。
* 外存地址：用于指出该页在外存上的地址，通常是物理块号，供调入该页时参考。

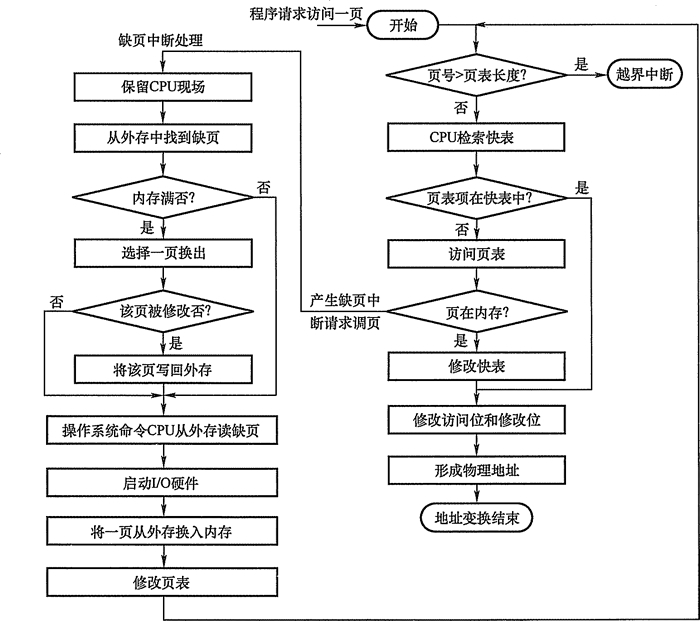
## 缺页中断机构

在请求分页系统中，每当所要访问的页面不在内存时，便产生一个缺页中断，请求操作系统将所缺的页调入内存。此时应将缺页的进程阻塞（调页完成唤醒)，如果内存中有空闲块，则分配一个块，将要调入的页装入该块，并修改页表中相应页表项，若此时内存中没有空闲块，则要淘汰某页（若被淘汰页在内存期间被修改过，则要将其写回外存)。  
  
缺页中断作为中断同样要经历，诸如保护CPU环境、分析中断原因、转入缺页中断处理程序、恢复CPU环境等几个步骤。但与一般的中断相比，它有以下两个明显的区别：

* 在指令执行期间产生和处理中断信号，而非一条指令执行完后，属于内部中断。
* 一条指令在执行期间，可能产生多次缺页中断。

## 地址变换机构

请求分页系统中的地址变换机构，是在分页系统地址变换机构的基础上，为实现虚拟内存，又增加了某些功能而形成的。

  
图3-25请求分页中的地址变换过程

如图3-25所示，在进行地址变换时，先检索快表：

* 若找到要访问的页，便修改页表项中的访问位（写指令则还须重置修改位)，然后利用页表项中给出的物理块号和页内地址形成物理地址。
* 若未找到该页的页表项，应到内存中去查找页表，再对比页表项中的状态位P，看该页是否已调入内存，未调入则产生缺页中断，请求从外存把该页调入内存。

# 3.8 页面分配策略：驻留集大小、调入页面的时机以及从何处调入页面

## 驻留集大小

对于分页式的虚拟内存，在准备执行时，不需要也不可能把一个进程的所有页都读取到主存，因此，操作系统必须决定读取多少页。也就是说，给特定的进程分配多大的主存空间，这需要考虑以下几点：

1. 分配给一个进程的存储量越小，在任何时候驻留在主存中的进程数就越多，从而可以提高处理机的时间利用效率。
2. 如果一个进程在主存中的页数过少，尽管有局部性原理，页错误率仍然会相对较高。
3. 如桌页数过多，由于局部性原理，给特定的进程分配更多的主存空间对该进程的错误率没有明显的影响。

基于这些因素，现代操作系统通常釆用三种策略：

1. 固定分配局部置换。它为每个进程分配一定数目的物理块，在整个运行期间都不改变。若进程在运行中发生缺页，则只能从该进程在内存中的页面中选出一页换出，然后再调入需要的页面。实现这种策略难以确定为每个进程应分配的物理块数目：太少会频繁出现缺页中断，太多又会使CPU和其他资源利用率下降。
2. 可变分配全局置换。这是最易于实现的物理块分配和置换策略，为系统中的每个进程分配一定数目的物理块,操作系统自身也保持一个空闲物理块队列。当某进程发生缺页时，系统从空闲物理块队列中取出一个物理块分配给该进程，并将欲调入的页装入其中。
3. 可变分配局部置换。它为每个进程分配一定数目的物理块，当某进程发生缺页时，只允许从该进程在内存的页面中选出一页换出，这样就不会影响其他进程的运行。如果进程在运行中频繁地缺页，系统再为该进程分配若干物理块，直至该进程缺页率趋于适当程度； 反之，若进程在运行中缺页率特别低，则可适当减少分配给该进程的物理块。

## 调入页面的时机

为确定系统将进程运行时所缺的页面调入内存的时机，可釆取以下两种调页策略：

1. **预调页策略。**根据局部性原理，一次调入若干个相邻的页可能会比一次调入一页更高效。但如果调入的一批页面中大多数都未被访问，则又是低效的。所以就需要釆用以预测为基础的预调页策略，将预计在不久之后便会被访问的页面预先调入内存。但目前预调页的成功率仅约50%。故这种策略主要用于进程的首次调入时，由程序员指出应该先调入哪些页。
2. **请求调页策略。**进程在运行中需要访问的页面不在内存而提出请求，由系统将所需页面调入内存。由这种策略调入的页一定会被访问，且这种策略比较易于实现，故在目前的虚拟存储器中大多釆用此策略。它的缺点在于每次只调入一页，调入调出页面数多时会花费过多的I/O开销。

## 从何处调入页面

请求分页系统中的外存分为两部分：用于存放文件的文件区和用于存放对换页面的对换区。对换区通常是釆用连续分配方式，而文件区釆用离散分配方式，故对换区的磁盘I/O速度比文件区的更快。这样从何处调入页面有三种情况：

1. 系统拥有足够的对换区空间：可以全部从对换区调入所需页面，以提髙调页速度。为此，在进程运行前，需将与该进程有关的文件从文件区复制到对换区。
2. 系统缺少足够的对换区空间：凡不会被修改的文件都直接从文件区调入；而当换出这些页面时，由于它们未被修改而不必再将它们换出。但对于那些可能被修改的部分，在将它们换出时须调到对换区，以后需要时再从对换区调入。
3. UNIX方式：与进程有关的文件都放在文件区，故未运行过的页面，都应从文件区调入。曾经运行过但又被换出的页面，由于是被放在对换区，因此下次调入时应从对换区调入。进程请求的共享页面若被其他进程调入内存，则无需再从对换区调入。

# 3.9 页面抖动(颠簸)和工作集(驻留集)

## 页面抖动（颠簸）

在页面置换过程中的一种最糟糕的情形是，刚刚换出的页面马上又要换入主存，刚刚换入的页面马上就要换出主存，这种频繁的页面调度行为称为抖动，或颠簸。如果一个进程在换页上用的时间多于执行时间，那么这个进程就在颠簸。  
  
频繁的发生缺页中断（抖动），其主要原因是某个进程频繁访问的页面数目高于可用的物理页帧数目。虚拟内存技术可以在内存中保留更多的进程以提髙系统效率。在稳定状态，几乎主存的所有空间都被进程块占据，处理机和操作系统可以直接访问到尽可能多的进程。但如果管理不当，处理机的大部分时间都将用于交换块，即请求调入页面的操作，而不是执行进程的指令，这就会大大降低系统效率。

## 工作集（驻留集）

工作集（或驻留集）是指在某段时间间隔内，进程要访问的页面集合。经常被使用的页面需要在工作集中，而长期不被使用的页面要从工作集中被丢弃。为了防止系统出现抖动现象，需要选择合适的工作集大小。  
  
工作集模型的原理是：让操作系统跟踪每个进程的工作集，并为进程分配大于其工作集的物理块。如果还有空闲物理块，则可以再调一个进程到内存以增加多道程序数。如果所有工作集之和增加以至于超过了可用物理块的总数，那么操作系统会暂停一个进程，将其页面调出并且将其物理块分配给其他进程，防止出现抖动现象。  
  
正确选择工作集的大小，对存储器的利用率和系统吞吐量的提嵩，都将产生重要影响。

# 3.10 内存管理知识点总结

分页管理方式和分段管理方式在很多地方相似，比如内存中都是不连续的、都有地址变 换机构来进行地址映射等。但两者也存在着许多区别，表3-20列出了分页管理方式和分段管理方式在各个方面的对比。

|  |  |  |
| --- | --- | --- |
| 表3-20 分页管理方式和分段管理方式的比较 | | |
|  | **分  页** | **分  段** |
| 目 的 | 页是**信息的物理单位**，分页是为实现离散分配方式，以消减内存的外零头，提髙内存的利 用率。或者说，分页仅权是由于系统管理的需要而不是用户的需要 | 是**信息的逻辑单位**，它含有一组其意义相对完整的信息。分段的目的是为了能更好地满足用户的需要 |
| 长 度 | 页的**大小固定且由系统决定**，由系统把逻辑地址划分为页号和页内地址两部分，是由机器硬件实现的，因而在系统中只能有一种大小的页面 | 段的长度不固定，**决定于用户所编写的程序**， 通常由编译程序在对流程序进行编译时，根据信息的性质来划分 |
| 地址空间 | **作业地址空间是一维的**，即单一的线性地址空间，程序员只需利用**一个记忆符，即可表示 一个地址** | **作业地址空间是二维的**，程序员在**标识一个地址时，既需给出段名，又需给出段内地址** |
| 碎 片 | 有**内部碎片  无外部碎片** | 有**外部碎片  无内部碎片** |
| ”共享“和“动态链接” | 不容易实现 | 容易实现 |