三、内存管理

一、考试大纲

（一）内存管理基础

1．内存管理概念

程序装入与链接；

逻辑地址与物理地址空间；

内存保护。

2．交换与覆盖

3．连续分配管理方式

单一连续分配；

分区分配。

4．非连续分配管理方式

分页管理方式；

分段管理方式；

段页式管理方式

（二）虚拟内存管理

1．虚拟内存基本概念

2．请求分页管理方式

3．页面置换算法

最佳置换算法（OPT）；

先进先出置换算法（FIFO）；

最近最少使用置换算法(LRU)；

时钟置换算法。

4．页面分配策略

5．抖动

抖动现象；

工作集

6．请求分段管理方式

7．请求段页式管理方式

二、知识点归纳

（一）内存管理基础

1．内存管理概念

内存是指处理器可以直接存取指令和数据的存储器，内存和处理器都是计算机系统的一种重要资源。在多道程序设计技术出现以后，对存储管理提出了更高的要求。内存管理的主要任务是为多道程序的运行提供良好的环境，方便用户使用存储器，提高存储器的利用率以

及从逻辑上扩充存储器。

在操作系统中，将一个用户的源程序变为一个可在内存中执行的进程，通常要经过以下几步：

1）编译。由编译程序将用户源代码编译成若干个目标模块；

2）链接。由链接程序将编译后形成的一组目标模块，以及它们所需要的库函数链接在一起，形成一个完整的装入模块；

3）装入。由装入程序将装入模块装入内存。

（1）程序的链接

源程序经过编译后，可得到一组目标模块，这时需要利用系统中链接程序将这组目标模块链接在一起，形成装入模块。根据链接时间的不同，可以分成三种链接方式，即静态链接、装入时的动态链接和运行时的动态链接。

1）静态链接。在程序运行前，先将各目标模块及它们所需的库函数，链接成一个完整的装配模块，以后不再拆开。静态链接方式需要解决两个问题：外部调用符号到相对地址的转换；相对地址到绝对地址的的转换；

2）装入时的动态链接。在将目标模块装入内存时，采用边装入边链接的方式。该方式具有以下优点：便于目标模块的修改和更新；便于实现对目标模块的共享。

3）运行时的动态链接。事先不对各个模块进行链接，而是在程序执行中当需要该模块时，再将该模块调入内存，并进行链接。采用这种方式，凡在执行过程中没有用到的目标模块，都不会被调入内存，这样不仅可加快程序的装入过程，而且可以节省大量的内存空间。

（2）程序的装入

程序的装入方式有三种，它们分别是绝对装入方式、可重定位的装入方式以及动态装入方式，现介绍如下。

1）绝对装入方式

在编译时，如果知道程序将驻留在内存的什么位置，那么编译程序将产生包含有内存绝对地址（也称实际地址）的目标代码。这样装入程序就会按照目标代码中的地址，将程序和数据装入到内存的实际位置。程序中使用的绝对地址，既可以在编译或汇编时给出，也可由程序员直接赋予。但因为需要程序员熟悉当前内存的使用情况，所以要求较高。

2）可重定位的装入方式

在多道程序的环境下，所得到的目标模块的起始地址通常是从 0 开始的，程序中的其它地址都是相对于起始地址进行计算的。此时应采用重定位装入方式，根据内存的当前情况，将装入模块装入到内存的适当位置。由于装入模块中的所有地址与实际装入内存的物理地址不同，所以需要进行地址变换（在原先相对地址的基础上加上内存装入位置的地址）。如下图所示

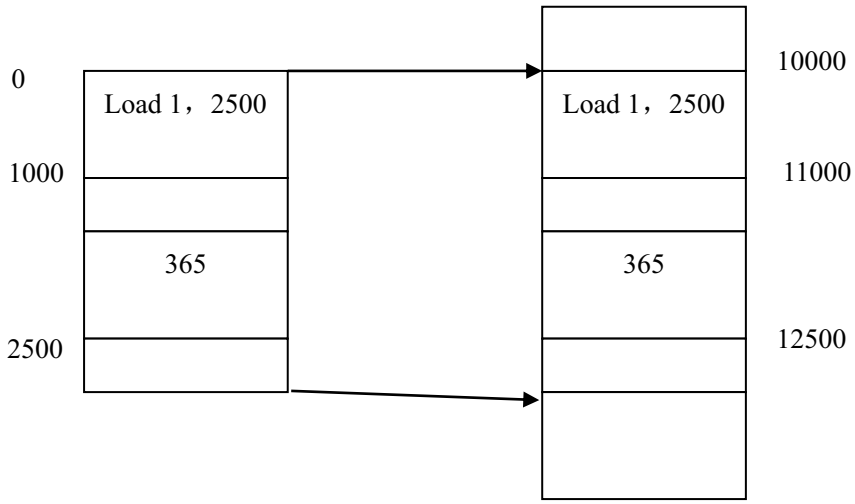

图 3-1 作业装入内存的情况

通常，把在装入程序时对目标模块中的指令和数据的地址变换过程称为重定位。又因为地址变换是在装入时一次完成的，所以称为静态重定位。

3）动态运行时的装入方式

动态运行时的装入程序，在把装入模块装入内存后，并不立即将其中的相对地址转换为绝对地址，而是把地址转换推迟到程序执行前再进行。因此，装入内存后的所有地址都仍是相对地址。

（3）逻辑地址与物理地址空间；

由编译程序所产生的所有目标模块中，使用的都是相对地址，即其起始地址为 0，每个模块中的地址都是相对于起始地址计算的。而物理地址是指内存中的实际的地址空间。

（4）内存保护

保证进入内存的各道作业都在自己的存储空间内运行，互不干扰。既要防止一道作业由于发生错误而破坏其他作业，也要防土破坏系统程序。这种保护一般由硬件和软件配合完成

2．交换与覆盖

交换技术就是把暂时不用的某个程序及数据部分（或全部）从内存移到外存中去，以便腾出必要的内存空间，或把指定的程序或数据从外存读到相应的内存中，并将控制权交给它，使其在系统上运行的一种内存扩充技术。

覆盖技术，就是把一个大的程序划分为一系列覆盖，每个覆盖就是一个相对独立的程序单位，把程序执行时并不要求同时装入内存的覆盖组成一组，称为覆盖段，这个覆盖段分配到同一个存储区域，这个存储区域称为覆盖区，它与覆盖段一一对应。

交换技术不要求程序员给出程序段之间的覆盖结构，而且交换主要是在进程或作业之间进行；而覆盖则主要在同一个作业或进程中进行。另外，覆盖只能覆盖与覆盖程序段无关的程序段。

3．连续分配管理方式

连续分配方式，是指为用户程序分配一个连续的内存空间。可分为单一连续分配方式和分区分配方式两大类。

（1）单一连续分配

这是一种最简单的存储管理方式，只能用于单用户、单任务的操作系统。采用该方式

时，可把内存分为系统区和用户区两部分，系统区位于内存的低址部分，供 OS 使用；用户区是除系统区以外的全部内存空间，是供用户程序使用的部分。

单一连续分配方式的主要特点是管理简单，只需要很少的软件和硬仍支持。且便于用户了解和使用。但采用这种存储分配方式，内存中只能装入一道作业运行，从而导致各类资源的利用率都不高。

（2）分区分配

分区式分配是能满足多道程序设计需要的一种最简单的存储管理技术。通常，按照分区的划分方式，它又可分为固定分区分配、动态分区分配以及可重定位的分区分配等三种方式。

1）固定分区分配

固定式分区法是一种最简单的可运行多道程序的存储管理方式。它将用户的内存空间划分为若干个固定大小的区域，在每个分区中只装入一道作业，这样，用户空间被划分为几个分区，便允许有几道作业并发运行。当有一个空闲分区时，便可以再从外存的后备作业队列中选择一个适当大小的作业装入该分区，当该作业结束时，又可再从后备作业队列中找出另一作业调入该分区。

可用下述两种方法将内存的用户空间划分为若干个固定大小的分区：

a）分区大小相等，就是使所有的内存分区大小相等。其缺点是缺乏灵活性，即当程序太小时，会造成内存空间的浪费；当程序太大时，一个分区又不足以装入该程序，致使该程序无法运行。尽管如此，这种划分方式仍被用于利用一台计算机去控制多个相同对象的场合，因为这些对象所需的内存空间是大小相等的。例如，炉温群控系统，就是利用一台计算机去控制多台相同的冶炼炉。

b）分区大小不等。为了克服分区大小相等而缺乏灵活性的这个缺点，把内存区划分成含有多个较小的分区、适量的中等分区及少量的大分区。这样，便可根据程序的大小为之分配适当的分区。

2）动态分区分配

动态分区分配是在作业装入和处理过程中建立的分区，要求分区的容量能正好适应作业的大小。在作业进入系统前，根据作业的大小来申请所需存储容量，然后由系统实施分配。

为了实现分区分配，系统中必须设置相应的数据结构，常用的有两种形式：空闲分区表和空闲分区链。

为把一个新作业装入内存，须按照一定的分配算法，从空闲分区表或空闲分区链中选出一分区分配给该作业。目前常用的有以下三种分配算法。

a）首次适应算法。以空闲分区链作为数据结构，在分配内存时，从链首开始顺序查找，直至找到一个大小能满足要求的空闲分区为止，从该分区中划出一块内存空间分配给请求者，余下的空闲分区仍留在空闲链中。如果从链首直至链尾都不能找到满足要求的分区，则此次内存分配失败。

b）循环首次适应算法。该算法是由首次适应算法演变而成的。在为进程分配内存空间时，不再是每次都从链首开始查找，而是从上次找到的空闲分区的下一个空闲分区开始查找。这种分配方式的优点是，使得内存中的空闲分区分布的更均匀，减少查找空闲分区的开销。

c）最佳适应算法。该算法要求将所有的空闲分区按其容量从小到大的顺序形成一个空闲分区链，这样，从链首开始查找时，第一次找到的合适分区，就是既能满足要求又是最小的空闲分区。

d)最坏适应算法

3）可重定位分区分配

在连续分配方式中，必须把一个作业装入连续的内存空间，时间长了，内存中会出现

很多内存碎片，而重定位式分区分配是解决碎片问题的简单而有效的办法。共基本思想是移动内存中的所有作业使之成为一个连续区域，继而留下一个较大的空白内存区。这种过程我们称之为“拼接”或“紧凑”。由于经过紧凑后的某些用户程序在内存中的位置发生了变化，所以还需要对程序和数据的地址进行修改，这叫做重定位。

4)分区的存储保护

①界限寄存器存储保护

a. 采用上、下界寄存器分别存放作业的结束地址和开始地址；

b. 采用基址和限长寄存器分别存放作业的起始地址及作业的地址空间长度。

②存储保护键：每个存储块分配一个独立的保护键，存储块不同于分区，一个分区由若干存储块组成，存储块大小相等，每个作业赋予一个保护键。

4．非连续分配管理方式

连续分配方式会形成许多“碎片”，虽然可通过“紧凑”方法将许多碎片拼接成可用的大块空间，但需要为之付出很大开销。如果允许将一个进程直接分散地装入到许多不相邻接的分区中，则无须再进行“紧凑”。基于这一思想而产生了离散分配方式。

（1）分页管理方式

所谓分页管理方式，是允许将一个进程直接分散地装入到许多不相邻接的分区中，分配的基本单位是页。

1）页面与页表

页面。分页存储管理是将一个进程的逻辑地址空间分成若干个大小相等的片，称为页面，并为各页加以编号。相应地，把内存空间也分成与页面相同大小的若干个存储块，称为(物理)块。在为进程分配内存时，以块为单位将进程中的若干个页分别装入到多个可以不相邻接的物理块中。一个页的地址分成两部分：页号和位移量，如下图

页表。在分页系统中，为了能在内存中找到每个页面所对应的物理块。系统又为每个进程建立了一张页面映像表，简称页表。在进程地址空间内的所有页，依次在页表中有一页表项，其中记录了相应页在内存中对应的物理号。如图 3-2 所示

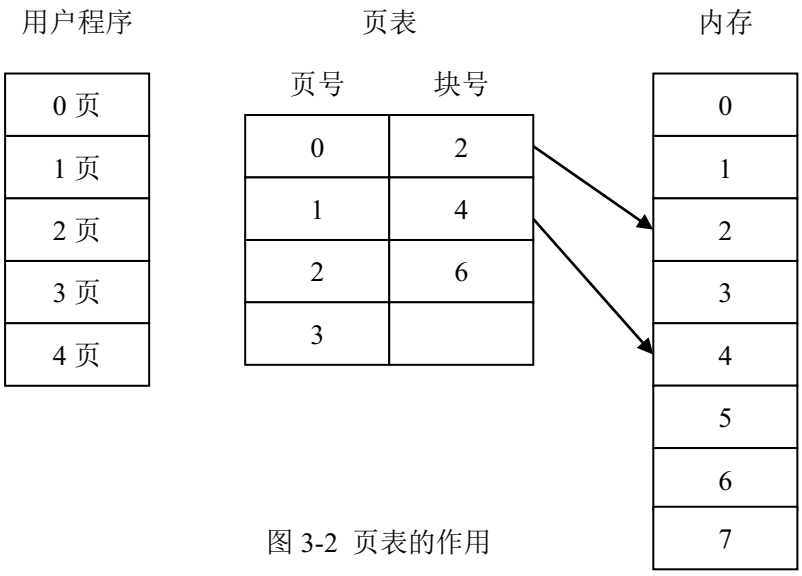

图 3-2 页表的作用

2）地址变换机构

为了能将用户地址空间中的逻辑地址变换为内存空间个的物理地址，在系统中必须设置地址变换机构。由于页内地址和物理地址是一一对应的，因此，地址变换任务就是借助于页表来完成的。

基本地址变换。在系统中只设置一个页表寄存器，用来存放页表在内存中的始址和页表的长度。当进程要访问某个逻辑地址中的数据时，分页地址变换机构会自动地将地址分为页号和页内地址两部分，再以页号为索引去检索页表。在检索前先将页号与页表长度相比较，判断是否有效。然后将页表始址与页号和页表项长度的乘积相加，便得到该表项在页表中的位置，进而得到该页的物理块号。

3）具有快表的地址变换机构

为了提高地址变换速度，可在地址变换结构中，增设一个具有并行查寻能力的特殊高速缓冲寄存器，称为快表。地址转换方式变为，将逻辑地址先在快表中查询转换，如失败，再访问内存中的页表。访问结束后，将该表项存入快表，以备下次查询。

4）两级页表和多级页表

现代的大多数计算机系统，都支持非常大的逻辑地址空间(2^{32} 一 2^{64})。在这样的环境下，页表就变得非常大，要占用相当大的内存空间。解决的方法有二级页表和多级页表。对于要求连续的内存空间存放页表的问题，可利用将页表进行分页，并离散地相各个页而分别存放在不同的物理块中的办法来加以解决，同样也要为离散分配的页表再建立一张页表，称为外层页表，在每个页表项中记录了页表页面的物理块号。为了地址变换实现上的方便起见，在地址变换机构中同样需要增设一个外层页表寄存器，用于存放外层页表的始址，并利用逻辑地址中的外层页号，作为外层页表的索引，从中找到指向页表分页的地址。

多级页表如果对外层页表再进行分页，并设置更高的页表来记录外层页表分页的存储情况。就构成了三级或多级页表。

（2）分段管理方式

所谓分段管理方式，是允许将一个进程直接分散地装入到许多不相邻接的分区中，分配的基本单位是段。

1）分段系统的基本原理

分段。在分段存储管理方式中，作业的地址空间被划分为若干个段，每个段定义了一

组逻辑信息。每段可用一个段号来代替段名，都是从 0 开始编址，并采用一段连续的地址空间。分段系统的的地址具有如下结构：

图 3-3 分段地址结

段表。像分页系统那样，在系统中为每个进程建立一张段映射表，简称“段表”。每个段在表中占用一个表项，其中记录了该段在内存中的起始地址和段的长度。

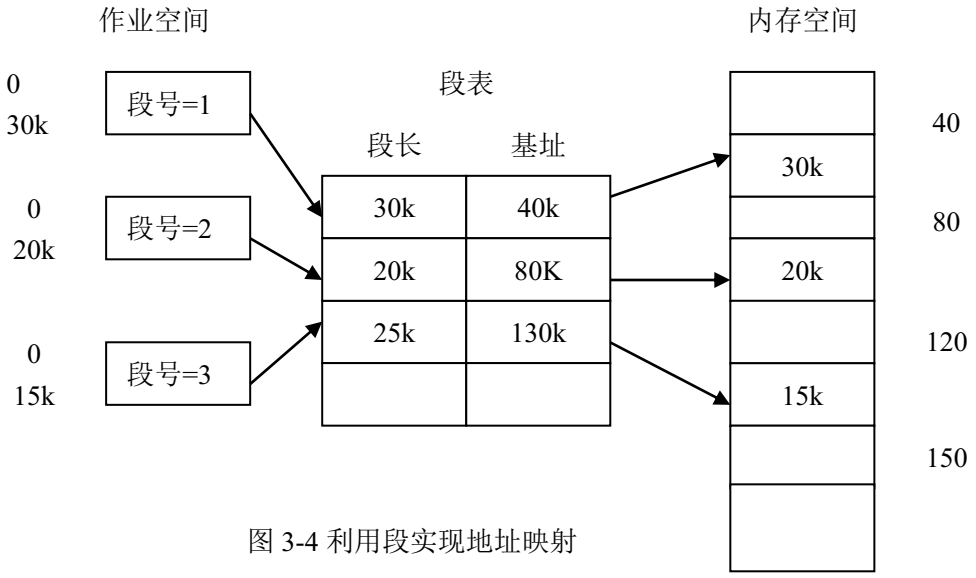

图 3-4 利用段实现地址映射

地址变换机构。为了实现从进程的逻辑地址到物理地址的变换功能，在系统中设置了段表寄存器，用于存放段表始址和段表长度。在进行地址交换时，系统将逻辑地址中的段号与段表长度进行比较。若段号大于段表长度，表示段号太大，访问越界，于是产生越界中断信号；若未越界，则根据段表的始址和该段的段号，计算出该段对应段表项的位置，从中读出该段内存的起始地址，然后，再检查段内地址是否超过该段的段长。若超过，同样发出越界中断信号；若未越界，则将该段的基址与段内地址相加，即可得到要访问的内存物理地址。

（3）段页式管理方式

分页与分段存储管理方式各有优缺点，所以将二者结合起来形成段页式存储系统，它既可以具有分段系统的便于实现、分段可共享、易于保护、可动态链接等优点，又可以像分页系统那样很好的解决内存的外部碎片问题。

基本原理。段页式系统先将用户程序分成若干个段，再把每个段分成若干个页，并为每个段取一个段名。如图 3-4 所示。

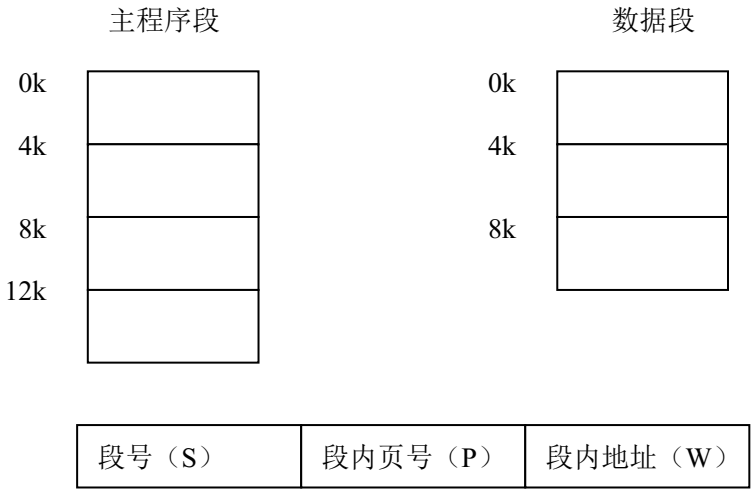

图 3-5 作业地址空间和地址结构

地址变换过程。在段页式系统中，为了便于实现地址变换，须设置一个段表寄存器，其中存放段表始址和段长。进行地址转换时，首先利用段号 S，将它与段长 TL 进行比较。若 S<TL，表示未越界，于是利用段表始址和段号求出该段所对应的段表项在段表中的位置，从中得到该段的页表地址，并利用逻辑地址中的段内页号 P 来获得对应页的页表项位置，从中读出该页所在的物理块号 b，再利用块号 b 和页内地址来构成物理地址。

（二）虚拟内存管理

1．虚拟内存基本概念

基于局部性原形，应用程序在运行之前，没有必要全部装入内存，仅需将那些当前需要运行的部分页面或段先装入内存便可运行，其余部分暂留在磁盘上。运行时，如其所要访问的页（段）已调入内存，便可继续执行下去；如未调入内存（称为缺页或缺段），此时程序利用操作系统所提供的请求调页（段）功能，将它们调入内存，以便进程能继续执行下去。如果此时内存已满，无法再装入新的页（段），则还须利用页（段）的置换功能，将内存中暂时不用的页（段）调至硬盘上，腾出足够的内存空间后，再将要访问的页（段）调入内存，使程序继续执行下去。这样，使可使一个大的用户程序能在较小的内存空间中运行，也可在内存中同时装入更多的进程使它们并发执行。从用户角度看，该系统所具有的内存空间 ，将比实际内存容量大得多，所以把这样的存储器系统称为虚拟存储器。

2．请求分页管理方式。在分页系统的基础上增加了请求调页功能和页面置换功能所形成的页式虚拟存储系统。

（1）硬件支持。主要的硬件支持有：

1）请求分页的页表机制。它是在纯分页的页表机制上增加若干项形成的数据结构。其基本作用是将程序地址中的逻辑地址转换成内存中的物理地址。如图 3-4 所示

页号	物理块号	状态位	访问字段	修改位	外存地址

图 3-6 扩充后的页表项

2）缺页中断机构。即每当用户程序要访问的页面尚未调入内存时，就产生缺页中断，请求操作系统根据该页在外存的地址把它调入内存。缺页中断是一个比较特殊的中断，它与

一般的中断相比，有着明显的区别，主要表现如下：在指令的执行期间产生和处理缺页中断；一条指令可以产生多个缺页中断。

3）地址变换机构。它同样是从分页地址变换机构的基础上发展形成的。

（2）实现请求分页的软件。这里包括有用于实现请求调页的软件和实现页面置换的软件。它们在硬件的支持下将程序运行时所需的页面调入内存，再将内存中暂时不用的页面从内存置换到磁盘上。

3．页面置换算法

页面置换算法是用来选择换出页面的算法。页面置换算法的优劣直接影响到系统的效率。

（1）最佳置换算法（OPT）

最佳置换算法是从内存中选择永远不再需要的页面或在最长的时间以后才需要访问的页面予以淘汰。实际上，这种算法无法实现，因为页面访问的未来顺序是很难精确预测的。但可以利用该算法评价其他算法的优劣。

（2）先进先出置换算法（FIFO）

先进先出置换算法总是选择在内存中驻留时间最长的页面予以沏汰，即先进入内存的页面先推出内存。这种方法的出发点是最早调入主存的页面，其不再使用的可能计会大一些。该算法对只有按线性顺序访问的程序比较合适，而对其他情况则效率不高。

（3）最近最少使用置换算法(LRU)

该算法选择最近一段时间内最长时间没有被访问过的页面予以淘汰。这种算法的主要出发点是，如果某个页由被访问过了，则它可能马上还要被访问到，或者反过来说，如果某页面很长时间未被访问，则它在最近一段时间也不会被访问。

（4）时钟置换算法。

也称为最近末使用算法，它是 LRU 和 FIFO 的折衷。该算法要求为每页设置一个访问位，并将内存中的所有页链接成一个循环队列。当某页被访问时，系统将其访问位设置为1。置换时采用一个指针，从当前的指针位置开始在、按序检查各页，若访问位为 0 则选择该页换出，若访问位为 1 则将其设置为 0，最后指针停留在被置换页的下一页上。

4．页面分配策略

1）固定分配局部置换。采用这种策略时，为每一个进程分配固定数量的物理块，在进程的整个运行期间进程拥有的物理块数不再改变。如果进程在运行中出现缺页，则只能从该进程的几个页面中选择一页换出，然后再调入该页，以保证分配给该进程的内存空间量不变。

2）可变分配全局置换。采用这种策略时，先为系统中的每一个进程分配一定数量的物理块，操作系统本身也保持一个空闲物理块队列。当某个进程发生缺页时，由系统从空闲物理块队列小取出一个物理块分配给该进程，并将缺页装入其中。这样，凡产生缺页的进程都获得新的物理块。

3）可变分配局部置换。采用这种策略时，为每一个进程分配一定数量的物理块，当某个进程发生缺页时，只允许从该进程的页面今选出一页换出，这样就不会影响其他进程的运行。如果某个进程在运行过程中频繁地发生缺页中断，则系统再为该进程分配若干个物理块，直到进程的缺页率降低到到适当程度为；反之，若一个进程在运行过程中的缺页率特别低.则系统可适当减少分配给该进程的物理块数。

5．抖动

抖动现象。刚被淘汰出内存的页面，过后不久又要访问它，需要再次将其调入，而该页调入内存后不久又再次被淘汰出内存，然后又要访问它。如此反复，使得系统把大部分时间用在了页面的调进换出上，而几乎不能完成任何有效的工作，这种现象被称为抖动。

工作集。指在某段时间间隔里，进程实际要访问的页面集合。虽然只要装入少数几页就

可以启动程序运行，但为使程序能有效地运行，较少地产生缺页，就必须使程序的工作集全部在内存中，然而，由于我们无法预知程序在不同时刻将访问哪些页面，因而只能像置换算法那样，利用程序过去某段时间内的行为，作为程序在将来某段时间内行为的近似。具体的说，便是把某进程在时间 t 的工作集记为 w（t，△），把变量△称为工作集窗口尺寸。

6．请求分段管理方式

这是在分段系统的基础上，增加了请求调段从分段置换功能后，所形成的段式虚拟存储系统。它允许只装入若干段(而非所有的段)的用户程序和数据，即可启动运行。以后再通过调段功能和段的置换功能，将暂不运行的段调出，同时调入即将运行的段。置换是以段为单位进行的。

为了实现请求分段，系统同样需要必要的破件支持。一般需要下列支持：

1）请求分段的段表机制。这是在纯分段的段表机制基础上增加若干项而形成的。如图 3-5 所示

段名	段长	段的基址	存取方式	访问方式	修改位	存在位 P	增补位

图 3-7 扩充后的段表项

2）缺段中断机构。每当用户程序所要访问的段尚末调入内存时，产生一个缺段中断，请求操作系统将所缺的段调入内存。

3）地址变换机构。请求分段系统的地址变换机构是在分段系统地址交换机构的基础上形成的。因为被访问的段并非全在内存，因而在地址变换时，如果发现所要访问的段不在内存时，必须先将所缺的段调入内存，并修改段表之后，才能再利用段表进行地址变换，为此，变换机制增加某些功能。如缺段中断的请求及处理等。

7．请求段页式管理方式

目前，有不少虚拟存储器是建立在段页式系统基础上的，通过增加请求页和页面置换功能而形成了段页式虚拟存储器系统，而且把实现虚拟存储器所需支持的硬件集成在处理器芯片上。

。