# 4.存储管理

存储器管理的对象是主存储器简称主存或内存。

存储管理的主要功能：主存空间的分配和回收、提高主存的利用率、扩充主存、对主存信息实现有效保护。

## 4.1存储管理的功能

### 4.1.1存储器的结构

存储器是计算机系统的重要资源之一。因为任何程序和数据以及各种控制用的数据结构都必须占用一定的存储空间，因此，**存储管理直接影响系统性能**。

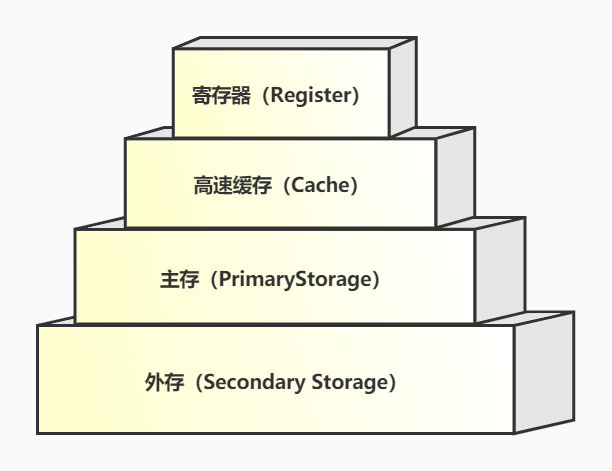
存储器由**内存**和**外存**组成。内存由**顺序编址的块**组成，**每块包含相应的物理单元**。CPU要通过**启动相应的输入输出设备后才能使外存与内存交换信息**。

**存储组织的功能：**在存储技术和CPU寻址技术许可的范围内组织合理的存储结构，使得各层次的存储器都处于均衡的繁忙状态。

**常用的存储器的结构：**“寄存器-主存-外存”结构和“寄存器-缓存-主存-存储组织的功能外存”结构。

**存储器系统：**是一个具有不同容量、成本和访问时间的存储设备的层次结构。**每一层于下一层相比都拥有较高的速度和较低延迟性，以及较小的容量**。

**存储器层次结构：**对应用程序的性能有着巨大的影响。理解系统是如何将数据在存储器层次结构中上下移动，可写出更符合系统运行的应用程序，运行更快。如下图：



图：存储器的层次结构

上图中，简单来说，每一层用来缓存下一层中的数据。

1. **虚拟地址**

1）虚拟地址介绍

对于程序员来说，数据的存放地址是由符号决定的，故称符号名地址，或者称为名地址，而把源程序的地址空间称为符号名地址空间或者名空间。它是从0号单元开始编址，并顺序分配所有的符号名所对应的的地址单元，所以它并不是主存中的真实地址，故称为相对地址、程序地址、逻辑地址或虚拟地址。

**符号名地址：**用符号标记的地址，访问时需要映射为实际内存单元的地址。就是一个虚拟地址。

2）[为什么要引入虚拟地址](https://blog.csdn.net/xiaolongwoaini99/article/details/100105504)

在早期的计算机中，要运行一个程序，会把这些程序全都装入内存，访问的内存地址都是实际的物理内存地址。当计算机同时运行多个程序时存在一些问题。

**问题1：**进程地址空间不隔离。由于程序都是直接访问物理内存，所以恶意程序可以随意修改别的进程的内存数据，以达到破坏的目的。有些非恶意的，但是有bug的程序也可能不小心修改了其它程序的内存数据。

**问题2：**内存使用效率低。程序要运行必须整个装入内存。

**问题3：**程序运行的地址不确定。当内存中的剩余空间可以满足程序C的要求后，操作系统会在剩余空间中随机分配一段连续的20M大小的空间给程序C使用，因为是随机分配的，所以程序运行的地址是不确定的。

为了解决上述问题，人们想到了一种变通的方法，就是增加一个中间层，利用一种间接的地址访问方法访问物理内存。按照这种方法，程序中访问的内存地址不再是实际的物理内存地址，而是一个虚拟地址，然后由操作系统将这个虚拟地址映射到适当的物理内存地址上。这样，只要操作系统处理好虚拟地址到物理内存地址的映射，就可以保证不同的程序最终访问的内存地址位于不同的区域，彼此没有重叠，就可以达到内存地址空间隔离的效果。

**2.地址空间**

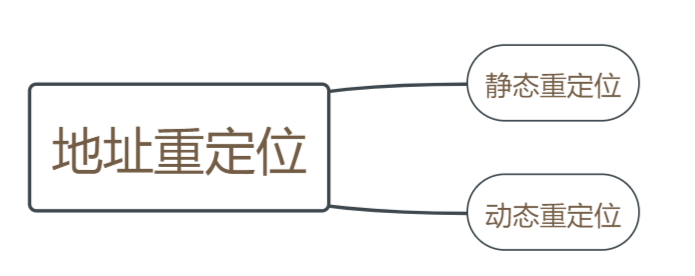
把程序中由符号名组成的空间称为名空间。源程序经过汇编或编译后再经过链接编辑程序加工形成程序的装配模块，即转换为相对地址编址的模块，它是以0为基址顺序进行编址的。相对地址也称为逻辑地址或虚地址，把程序中由相对地址组成的空间称为逻辑地址空间。相对地址空间通过地址再定位机构转换到绝对地址空间，绝对地址空间也叫物理地址空间。

当创建一个进程时，操作系统会为该进程分配一个4GB大小的虚拟进程地址空间。之所以是4GB，是因为在32位的操作系统中，一个指针长度是4字节，而4字节指针的寻址能力是从0x00000000~0xFFFFFFFF，最大值0xFFFFFFFF表示的即为4GB大小的容量。与虚拟地址空间相对的，还有一个物理地址空间，这个地址空间对应的是真实的物理内存。如果你的计算机上安装了512M大小的内存，那么这个物理地址空间表示的范围是0x00000000~0x1FFFFFFF。

**3.存储空间**

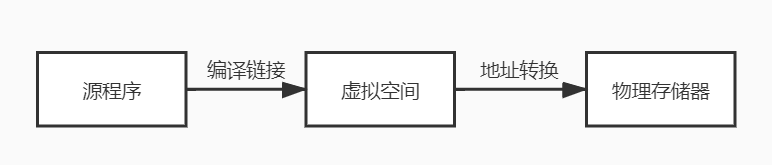
简单来说，逻辑地址空间（简称地址空间）是逻辑地址的集合，物理地址空间（简称存储空间）是物理地址的集合。

### 4.1.2地址重定位



**定义：**地址重定位是指将逻辑地址变换成主存物理地址的过程。

用于在可执行文件装入时，需要解决可执行文件中地址（指令和数据）与主存地址的对应关系。



图：地址变换

地址重定位由操作系统中的**装入程序Loader**和**地址重定位机构**来完成。

地址重定位分为**静态地址重定位**和**动态地址重定位**。

**1.静态重定位**

静态重定位是指**在程序装入主存时**已经完成了逻辑地址到物理地址的变换，在程序的执行期间将不会再发生变化。早期的操作系统采用此方法。

**优点：**

无须硬件、地址变换机构的支持，只要求程序本身是可重定位的，只对那些要修改的地址部分具有某种标识，由专门设计的程序来完成。

**缺点：**

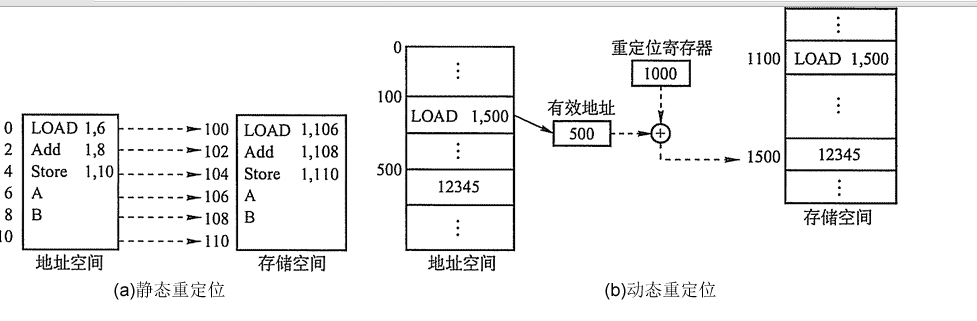
必须给作业分配一个连续的存储区域，在作业的执行期间不能扩充存储空间，也不能在主存中移动，多个作业也难以共享主存中的同一程序副本和数据。

**2.动态重定位**

动态重定位是指在**程序运行期间**完成逻辑地址到物理地址的变换。其实现机制要依赖硬件地址变换机构，如基地址寄存器（BR）。

**优点：**

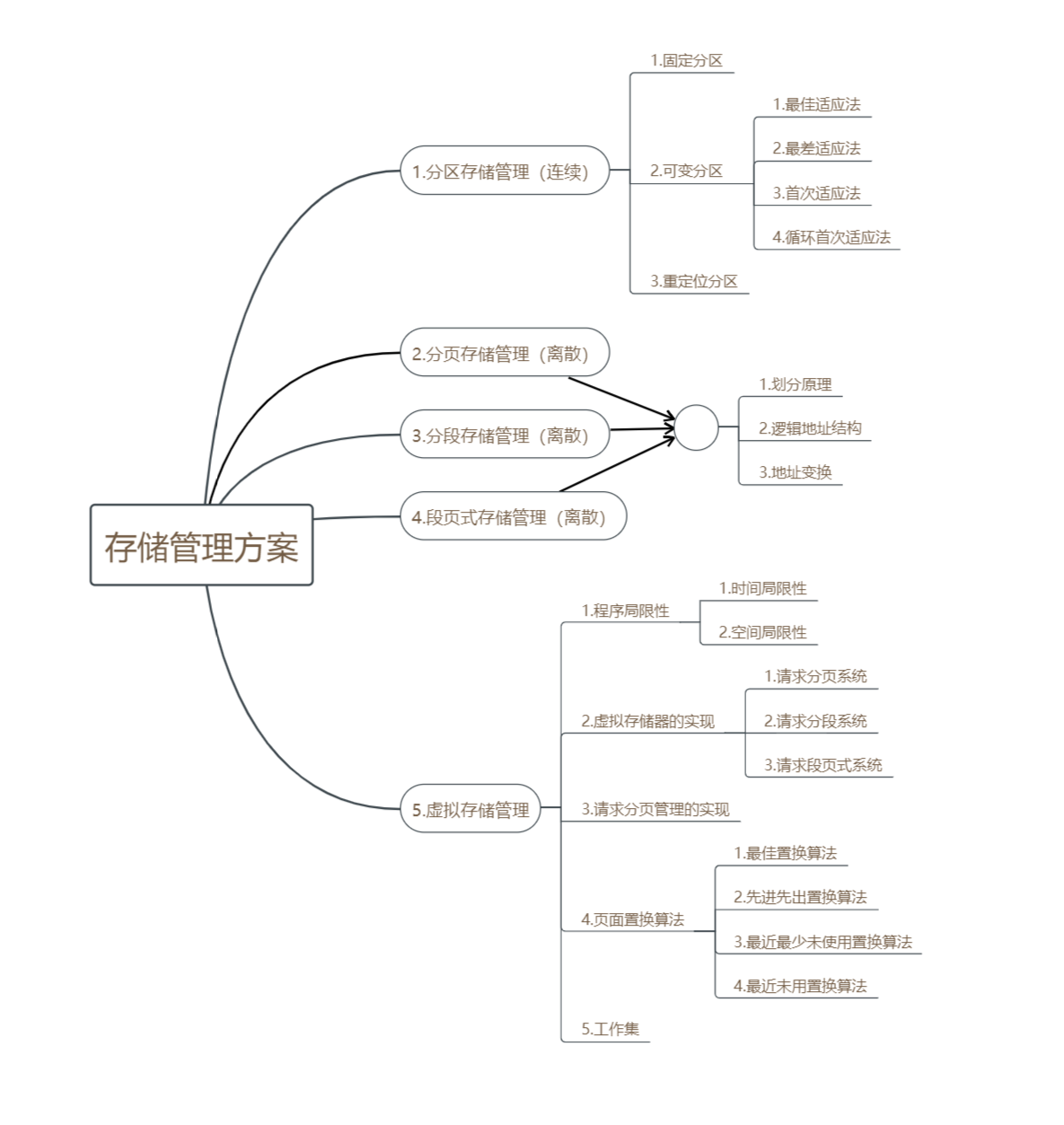
程序在执行期间可以换入和换出内存，以解决主存空间不足的问题；可以在主存中移动，把主存中的碎片集中起来，以充分利用空间；不必给程序分配连续的主存空间，可以较好地利用较小的主存快；可以实现共享。



## 4.2存储管理方案

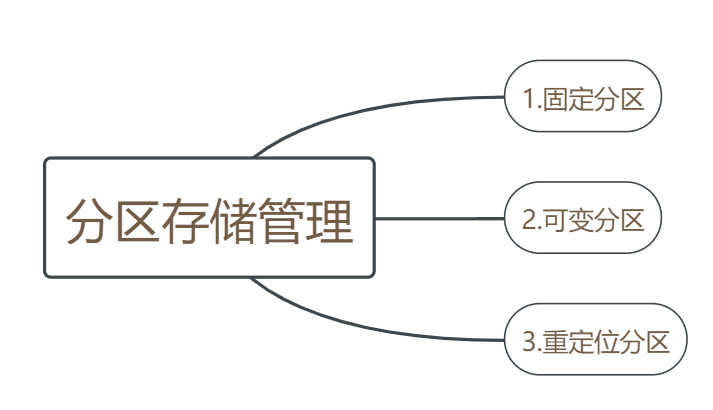
**存储管理的主要目的：**解决多个用户使用主存的问题。

存储管理方案主要包括**分区存储管理** 、**分页存储管理**、**分段存储管理**、**段页式存储管理**以及**虚拟存储管理**。



### 4.2.1分区存储管理

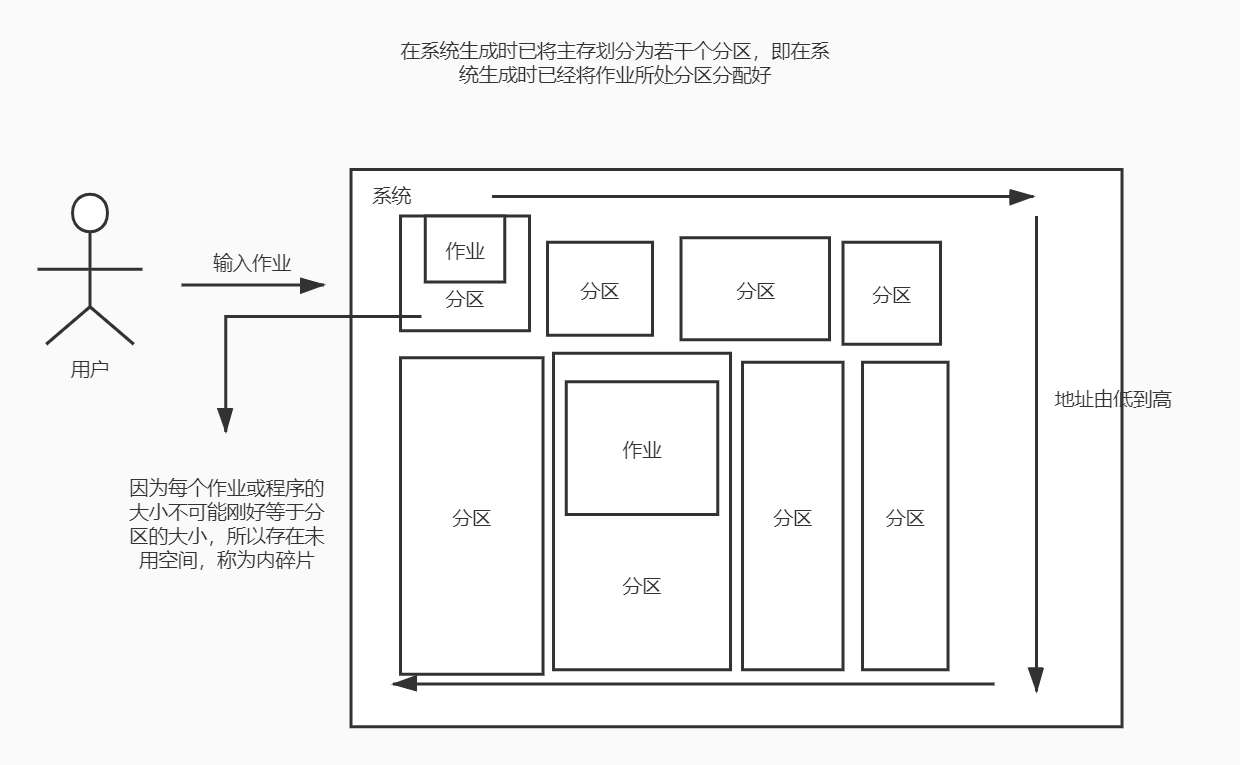
**定义：**早期的存储管理方案，基本思想是把主存的用户区划分成若干个区域，**每个区域分配给一个用户作业使用**，并限定它们只能在自己的区域中运行，这种主存分配方案就是分区存储管理方式。



#### 1.固定分区

**原理：**是一种**静态**分区方式，在系统生成时已将主存划分为若干个分区，每个分区的大小可不等。操作系统通过主存分配情况表管理主存。

**缺点：**已分配区中存在未用空间。因为程序或作业的大小不可能刚好等于分区的大小，故造成了空间的浪费。（将已分配区未使用空间称为零头或**内碎片**）



图：固定分区

#### 2.可变分区

**原理：**是一种动态分区方式，存储空间的划分是在作业装入时进行的，故分区的个数是可变的，分区的大小刚好等于作业的大小。可变分区分配需要两种管理表格，**已分配表**记录已分配分区的情况，**未分配表**记录未分配分区的情况。

**优缺点：**可变分区使主存分配更灵活，提高主存利用率，但由于系统不断的分配和回收，会出现不连续的小空闲区，由于不连续而无法分配，产生了未分配的无用空间，即外碎片。解决外碎片的方法是拼接（紧凑），即向一个方向（如向低地址端）移动已分配的作业，是那些外碎片再另一个方向（即高地址段）连成一片。

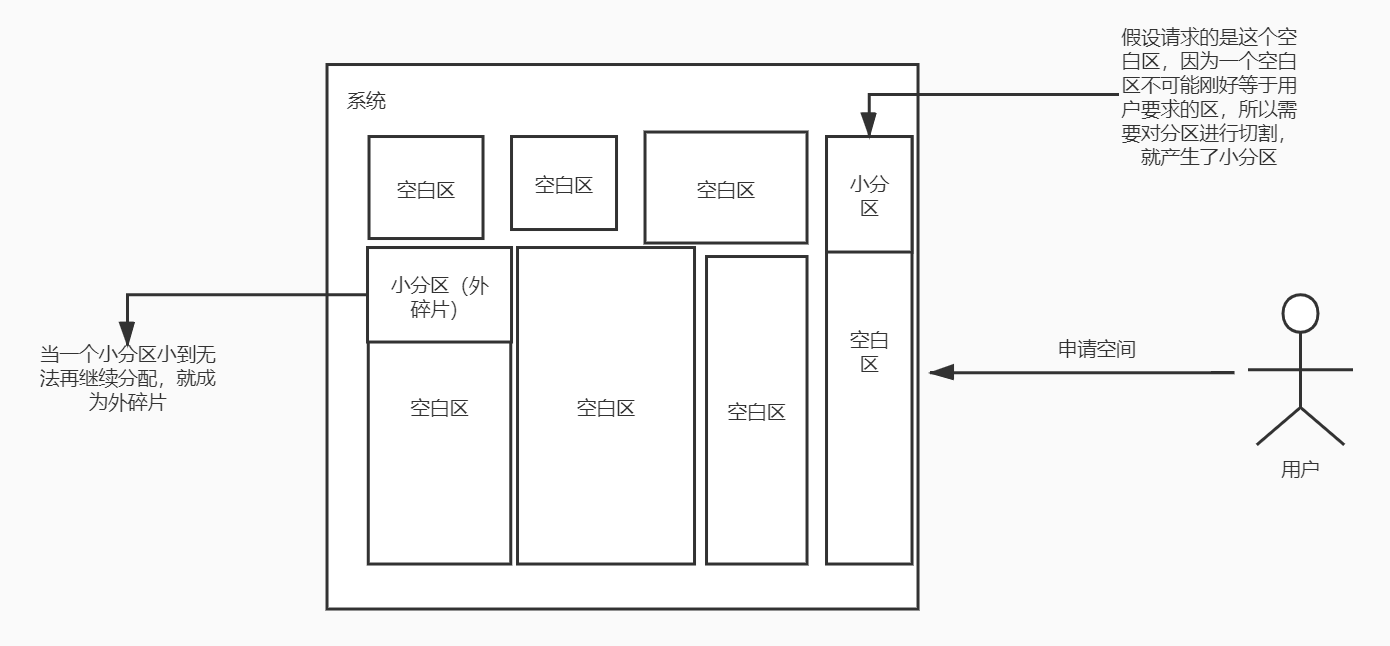
对于可变分区的请求和释放分区需要相应的算法，主要有以下四种算法：

**1).最佳适应算法**

系统中有n个空白区（自由区），每当用户申请一个空间时，将从这个空白区中找到一个最接近用户需求的空白分区，但空闲区不可能刚好等于用户需要的区的大小，所以必然要将一个分区一分为二，但随着系统不断的释放空间（分区不断的一分为二），会使产生的小分区小到无法再继续分配，将这样无用的分区称为**外碎片。**

优点：能保留较大的空白区

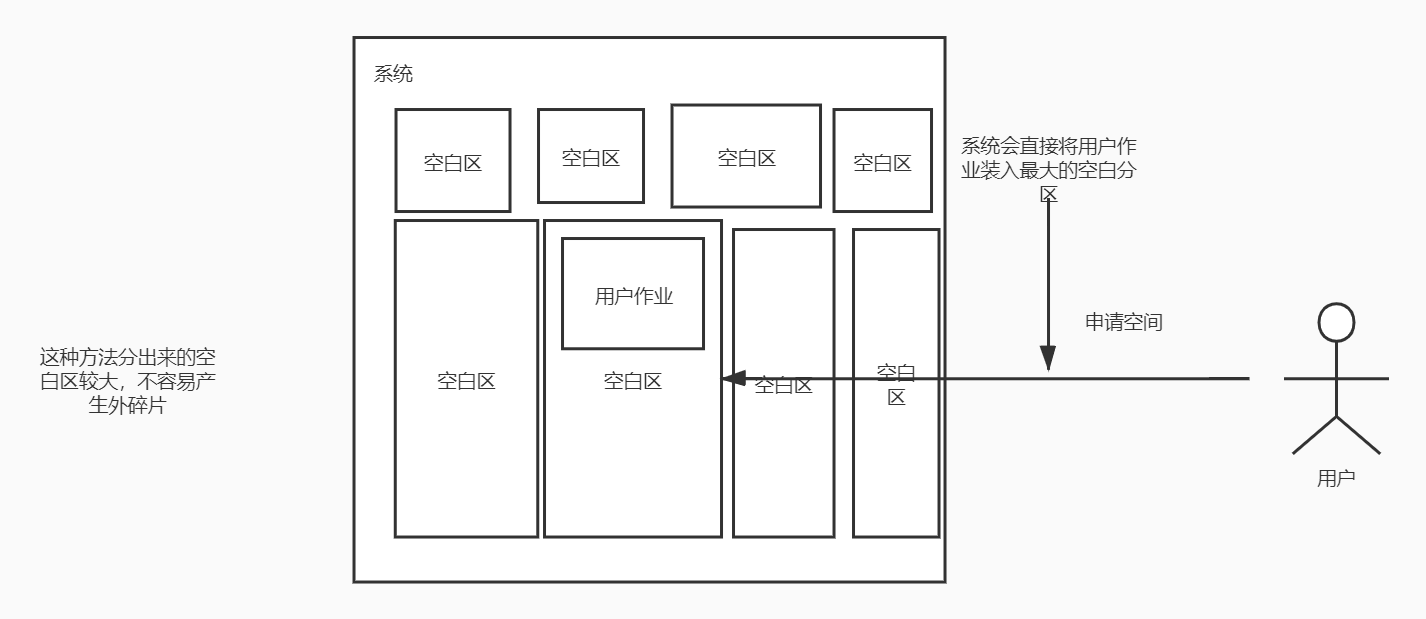
缺点：会产生过小的分区即外碎片



图：最佳适应算法

**2).最差适应算法**

系统总是将用户作业装入最大的空白分区。这种算法将一个最大的分区一分为二，所以剩下的空白区通常也大，不容易产生外碎片。



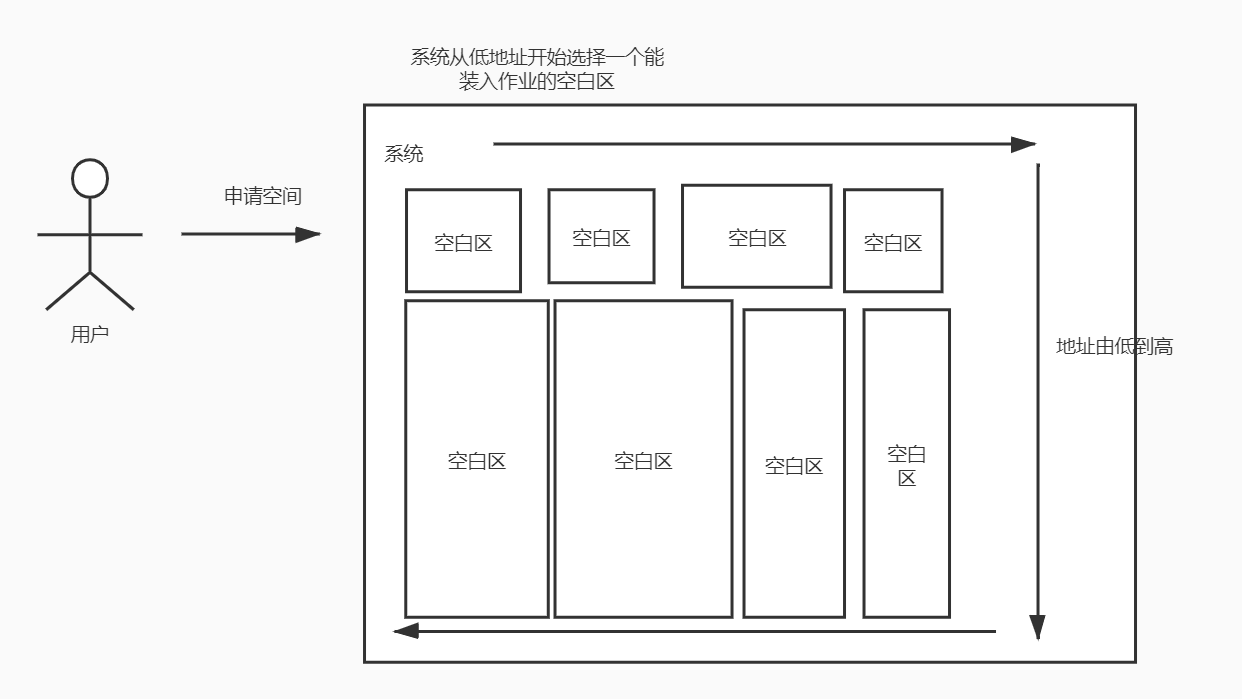
图：最差适应算法

**3).首次适应算法**

每当用户作业申请一个空间时，系统总是从主存的低地址开始选择一个能装入作业的空白区。

优点：用户释放空间时，更容易实现相邻的空白区合并。

缺点：每次查找又都从低地址部分开始，会增加查找的开销。

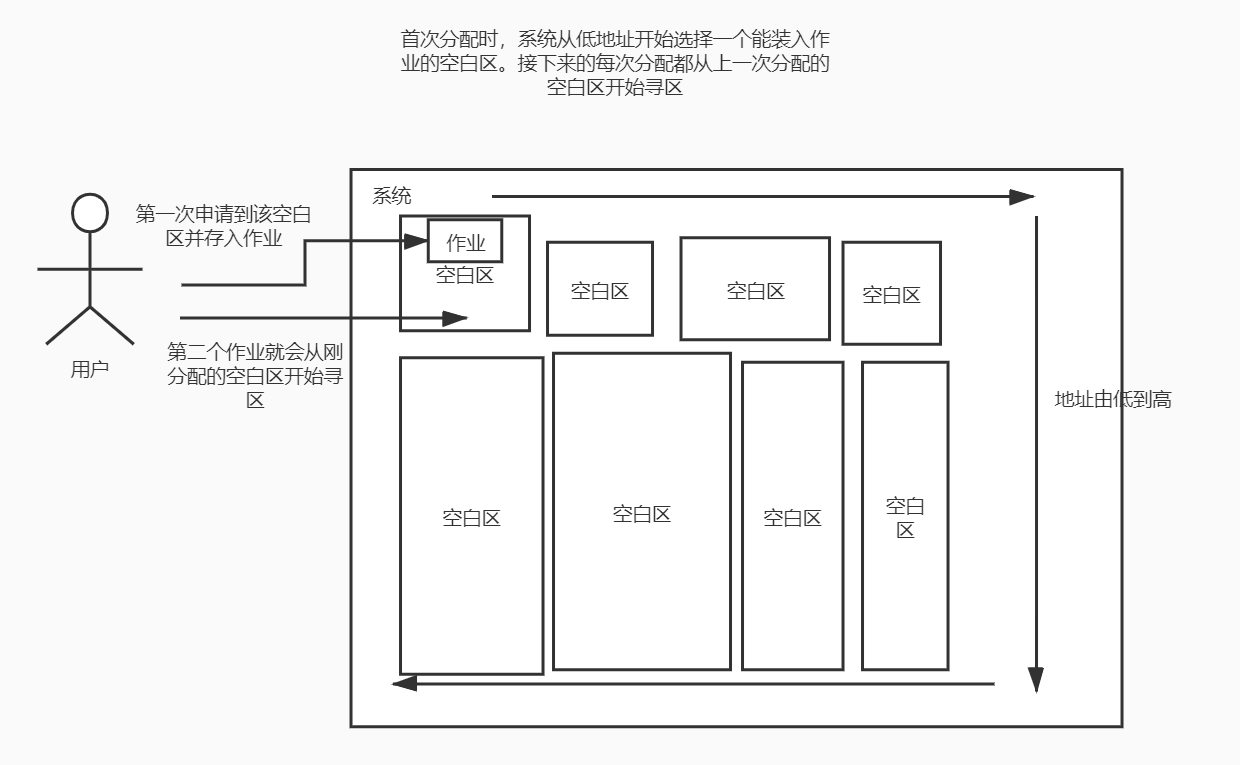


图：首次适应算法

**4).循环首次适应算法**

与首次适应算法的不同之处是，每次分配都是从刚分配的空白区开始寻找一个能满足用户要求的空白区。

优点：使内存中的空闲分区分布的更为均匀，减少了查找时的系统开销。



图：循环首次适应算法

**可变分区不可避免得会产生外碎片问题。**

**解决外碎片的方法：**拼接，即向一个方向（例如向低地址端）移动已分配的作业，使那些零散的小空闲区在另一个方向连成一片。

#### 3.可重定位分区

可重定位分区是**解决碎片问题**的简单且行之有效的方法。

**基本思想：**移动所有已分配好的分区，使之成为连续区域。如同队列有一个队员出列，指挥员要求大家“靠拢”一样。

**执行时机：**分区“靠拢”的时机是当用户请求空间得不到满足时或某个作业执行完毕时。由于靠拢是要代价的，所以通常是在用户请求空间得不到满足时进行。

需要注意的是，当进行分区“靠拢”时会导致地址发生变化，所以有地址重定位问题，需要地址重定位技术的支持。

#### 4.分区保护

---- 存储保护是为了防止一个作业有意或无意地破坏操作系统或其他作业。

1）上、下界寄存器方法。采用上、下界寄存。；器分别存放作业的结束地址和开始地址。在作业运行过程中，将每一个访问内存的地址都同这两个寄存器的内容进行比较。在正常情况下，这个地址应大于下界寄存器，且小于上界寄存器的内容，如超出这个范围便产生保护性中断。

2）基址、限长寄存器方法。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。当作业执行时，将每一个访问内存的相对地址和这个限长寄存器比较，如果超过了限长，则发出越界中断信号，并停止作业的运行。

### 4.2.2分页存储管理

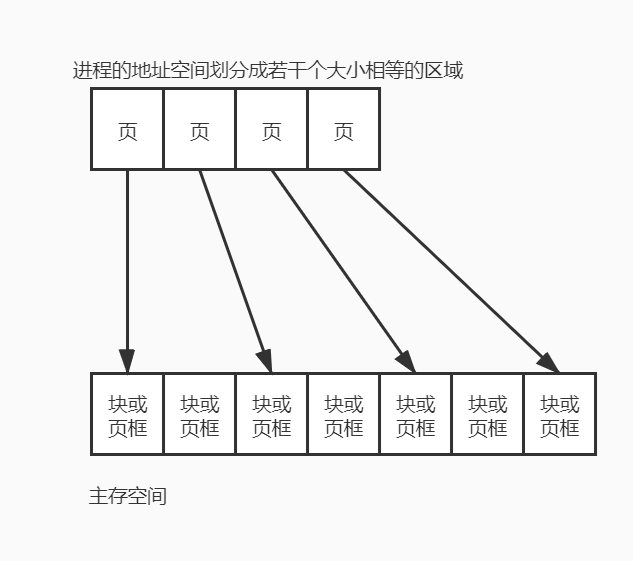
为什么不用分区管理了：分区管理解决了多道程序共享主存的可行方案，但用户程序必须装入连续的地址空间中，若无满足用户的连续空间，需要进行分区靠拢操作，这需要耗费系统时间为代价。因此引入分页管理。

#### 1.纯分页存储管理

**1分页原理**

**页的定义：**将一个进程的地址空间划分成若干个大小相等的区域，称为页面或**页**。

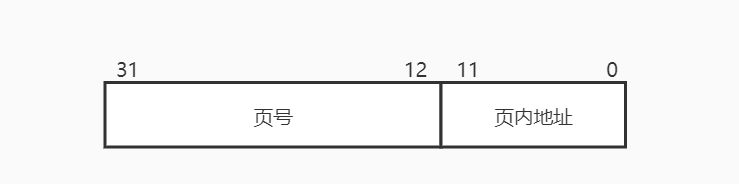
**块的定义：**相应地，将主存空间划分成与页相同大小的若干个物理块，称为**块**或页框。

在为进程分配内存时，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。由于进程的最后一页经常装不满一块而形成了不可利用的碎片，称之为**页内碎片**。

图：分页原理

**2.地址结构**

分页系统的地址结构如下图所示：



图：分页地址结构

**地址结构组成：**由两部分组成，前一部分为页号P;后一部分为偏移量W，即页内地址。图中的地址长度为32位，其中，0-11位为页内地址（每页的大小为212=4kb），12-31位为页号220=1Mb，所以允许地址空间的大小最多为1Mb个页。

**3.页表**

**页表意义：**

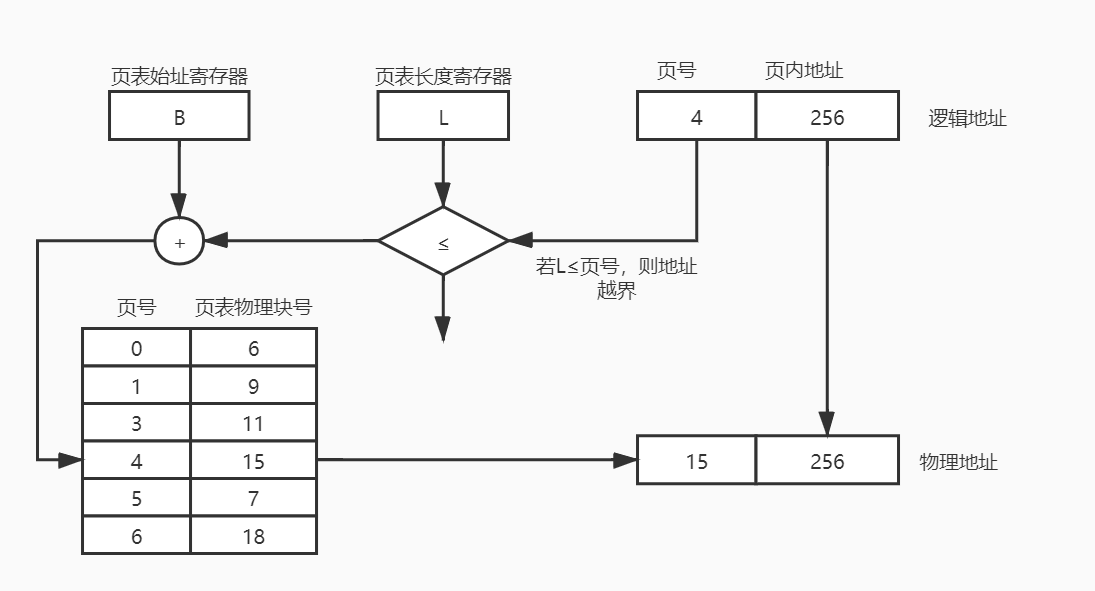
当进程的多个页面离散地分配到主存的多个物理块时，系统应能保证在主存中找到进程要访问的页面所对应的物理块。为此，系统为每个进程建立了一张页面映射表，简称页表。每个页在表中占一个表项，记录该页在主存中对应的物理块号。

**页表作用：**

实现从页号到物理块号的地址映射。

**如何实现及页表寄存器：**

进程未执行时，页表的始址和页表长度存放在本进程的PCB（进程控制块）中。当调度程序调度到某进程时，才将这两个数据装入**页表寄存器**中。在单处理机环境下，虽然系统中可以运行多个进程，但只需一个页表寄存器。



图：页式存储管理的地址映射

**实例分析：**

例如，进程在执行时，系统通过查找页表就可以找到每页所对应的物理块号。图中逻辑页号为4，查找页表可得该页的物理块号为15，与页内地址256拼接得到物理地址。可见，页表的作用是实现从页号到物理块号的地址映射。

**地址变换机构的基本任务：**

利用页表把用户程序中的逻辑地址变换成主存中的物理地址，实际上就是将用户程序中的页号变换成主存中的物理块号。

#### 2快表

**1.快表的由来**

由于页表是存放在内存中的，这使CPU在每存取一个数据时，都要两次访问内存。第一次是访问内存中的页表，从中找到指定页的物理块号，再将块号与页内偏移量W拼接，以形成物理地址。第二次访问内存时，才是从第一次所得地址中获得所需数据(或向此地址中写入数据)。

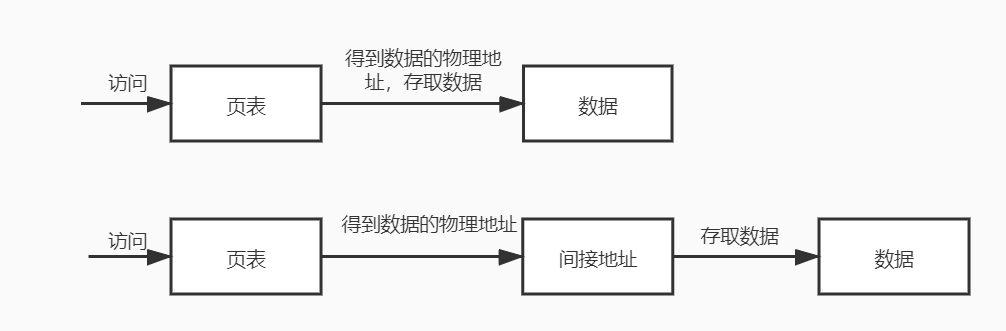
为了提高访问主存的速度有以下两种解决方案：

1）在地址变换机构中增设一个具有并行查寻能力的特殊高速缓冲寄存器，又称为“联想寄存器”，用来存储页表。此方法需要大量硬件开销，经济上不可行。

2）在地址变换机构中增设一个小容量的高速存储器，又称“联想存储器”，也称之为快表，用来存放当前访问频率高的少数活动页的页号及相关信息。

**2. 快表工作原理**

查找快表和查找内存页表是并行的，一旦在快表中找到相符的逻辑页号，就停止查找内存页表；若快表中找不到，则继续在内存的页表中查找，得到物理地址后，需将物理块号填入快表的空闲单元，若无空闲单元，则根据淘汰算法淘汰某一页，再填入新得到的页号。



图：地址映射访问内存过程

#### 3两级页表机制

**1两级页表机制的由来**

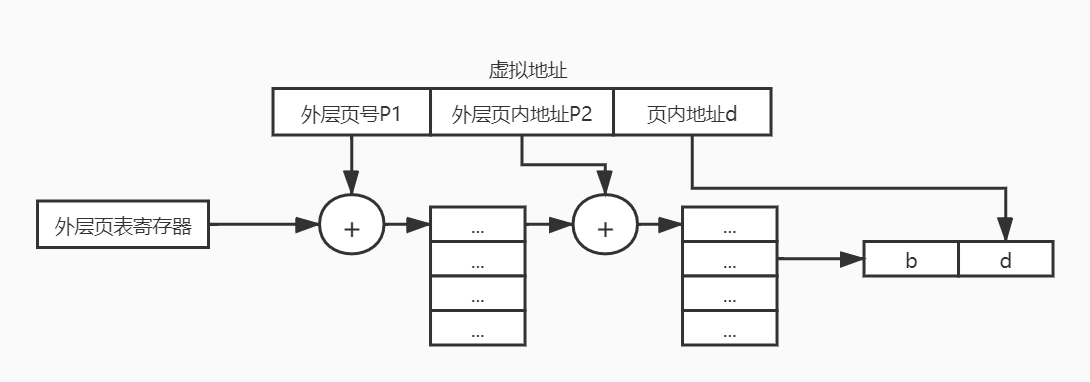
由80386的分页地址结构可知，每个表项的占用空间为32位即4B，且页表项（页号）达1Mb个，所以每个进程的页表占用4MB主存空间，并且页表的存储空间要求是连续的。为了减少页表所占用的连续的主存空间，在80386中采用了**两级页表机制**。

**两级页表优点：**为了减少页表所占用的连续的主存空间。

**基本方法：**

是将页表进行分页，每个页面的大小与主存物理块的大小相同，并为它们进行编号，可以离散地将各个页面分别存放在不同的物理块中。

为此需要建立一张页表，称为外层页表（页表目录），即第一级是页目录表，其中的每个表目是存放某个页表的物理地址；第二级是页表，其中的每个表目所存放的是页的物理块号。



图：两级页表的地址变换机构

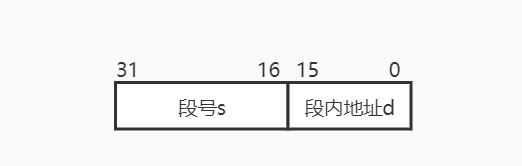
### 4.2.3分段存储管理

**1分段存储管理简介：**

将一个**作业**的地址空间被划分为若干个段，每个段是**一组完整的逻辑信息**，例如有主程序段、子程序段、数据段以及堆栈段等，每个段都有自己的名字，都是**从0开始编址**的一段连续的地址空间，各段的长度是不等的。段与段之间的地址不连续。

**2地址结构：**

由**段号（名）**和**段内地址**两部分组成。



图：分段的地址结构

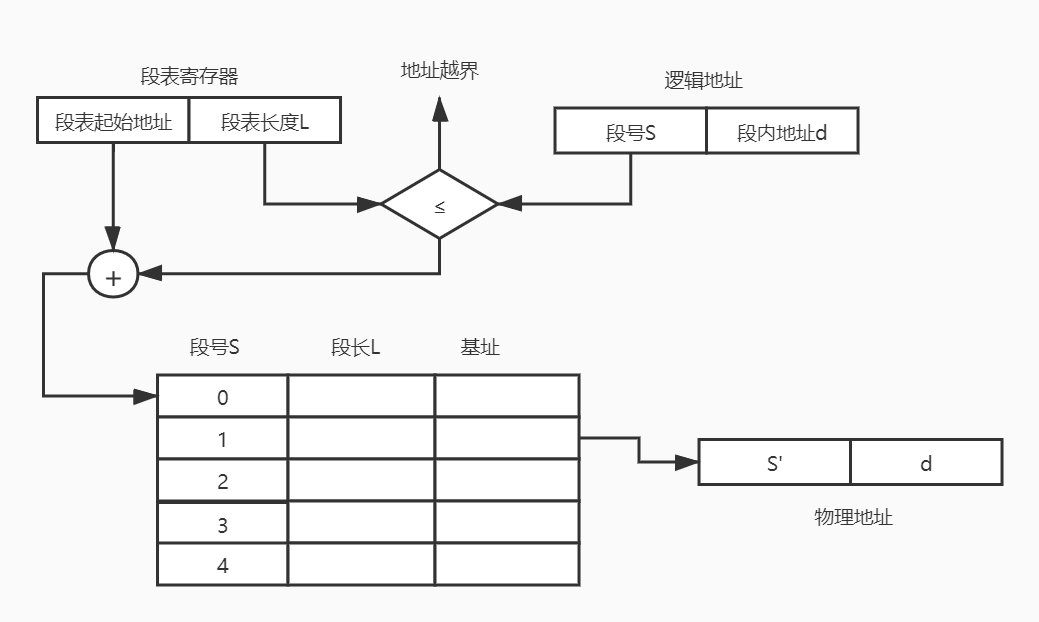
**3段表：**

在系统中为每个进程建立一张段映射表，简称为“段表”。每个段在表中占有一个表项，在其中记录了该段在主存中的起始地址（又称为“基址”）和段的长度。

进程在执行时，**通过查段表来找到每个段所对应的主存区。**

**4地址变换实现过程：**

为实现从逻辑地址到物理地址的变换功能，系统中设置了段表寄存器，用于存放段表始址和段表长度。在进行地址变换时，系统将逻辑地址中的段号S与段表长度L进行比较，若S≥L表示段号太大，访问越界。



图：段式存储管理的地址变换机构

### 4.2.4段页式存储管理

#### 1. 分页分段比较

分段和分页其实都是一种对地址的划分或者映射的方式。 h两者的区别主要有以下几点：

**1.**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，提高内存的利用率；或者说，**分页仅仅是由于系统管理的需要**，而不是用户的需要（也是**对用户透明**的）。**段是信息的逻辑单位，它含有一组其意义相对完整的信息**（比如数据段、代码段和堆栈段等）。**分段的目的是为了能更好的满足用户的需要**（用户也是可以使用的）。

**2.**页的大小固定且由系统确定，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，因而一个系统只能有一种大小的页面。**段的长度却不固定，决定于用户所编写的程序**，通常由编辑程序在对源程序进行编辑时，根据信息的性质来划分。

**3.**分页的作业地址空间是维一的，即单一的线性空间，程序员只须利用一个记忆符（线性地址的16进制表示），即可表示一地址。分段的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（比如数据段、代码段和堆栈段等），又需给出段内地址。

**4.**页和段都有**存储保护机制**。但存取权限不同：段有读、写和执行三种权限；而页只有读和写两种权限

#### 2分页和分段存储管理优缺点。

**分页优点：**因为分页的过程是由操作系统完成的，对用户是透明的，所以用户不必关心分页的过程。

**分页缺点：**不易实现共享。段是信息的逻辑单位。

**分段优点：**易于实现段的共享，即允许若干个进程共享一个或多个段，而且对段的保护也十分简单。

#### 3段页式存储管理优点

既具有分页系统能有效地提高主存利用率的优点，又具有分段系统能很好地满足用户需要的长处，显然是一种比较有效的存储管理方式。

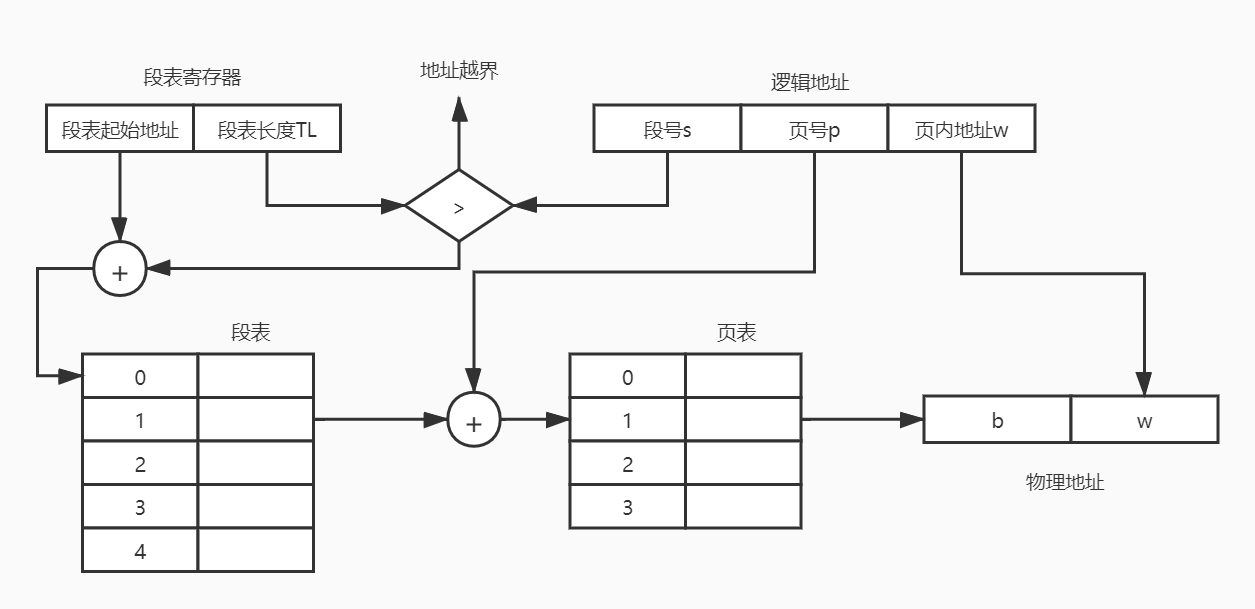
#### 4段页式系统基本原理

先将整个主存划分成大小相等的存储块（页框），将用户程序按程序的逻辑关系分为若干个段，并为每个段赋予一个段名，再将每个段划分成若干页。以页框为单位离散分配。在段页式系统中，其地址结构由段号、段内页号和页内地址三部分组成。



图：段页式管理的地址结构

#### 5地址变换实现过程

为实现从逻辑地址到物理地址的变换，系统中必须**同时配置段表和页表**。由于将段中的页进行离散地分配，段表中的内容不再是段的主存始址和段长，而是页表始址和页表长度。在段页式系统中有一个段表寄存器，用于存放段表起始地址和段表长度TL。

图：段页式存储管理的地址变换结构

**逻辑地址到物理地址变换过程如下：**

1. 根据段号S查段表，得到页表的起始地址；
2. 根据页号P查页表，得到物理块号b；
3. 将物理块号b拼页内地址w得到物理地址。

### 4.2.7虚拟存储管理

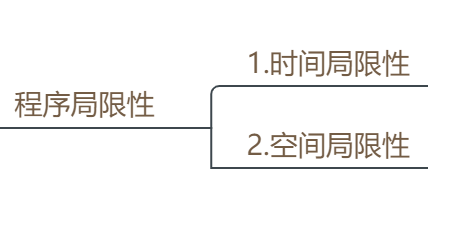
前面介绍的几种存储管理方案中，必须为每个作业分配足够的空间，以便装入全部信息。当主存空间不能满足作业要求时，作业无法装入主存执行。

#### 1虚拟存储器

**定义：**如果一个作业只部分装入主存便可开始启动运行，其余部分暂时留在磁盘上，在需要时再装入主存，这样可以有效地利用主存空间。**从用户角度看**，该系统所具有的主存容量将比实际主存容量大得多，人们把这样的存储器称为虚拟存储器。

**主要用处：**扩大主存容量。其容量是由计算机的地址结构决定的。

#### 2程序局部性原理



程序在执行时将呈现出局部性规律，即在一段时间内，程序的执行仅局限于某个部分。相应地，它所访问的存储空间也局限于某个区域内。程序的局限性表现在**时间局限性**和**空间局限性**两个方面。

**1.时间局限性**

指如果程序中的某条指令一旦执行，则不久的将来该指令可能再次被执行；如果某个存储单元被访问，则不久以后该存储单元可能再次被访问。

**产生原因：**程序中存在着大量的循环操作。

**2.空间局限性**

指一旦程序访问了某个存储单元，则在不久的将来，其附近的存储单元也最有可能被访问。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内。

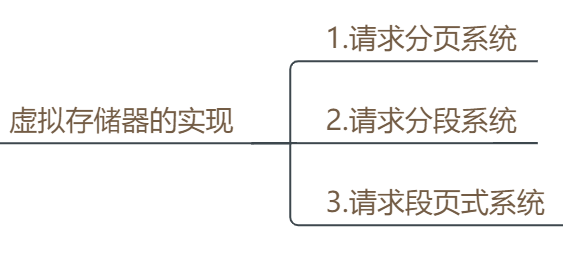
**产生原因：**程序是顺序执行的。

#### 3虚拟存储器的实现

**虚拟存储器**是具有请求调入功能和置换功能，能仅把作业的一部分装入主存便可运行作业的存储器系统，是能从逻辑上对主存容量进行扩充的一种虚拟的存储器系统。

虚拟存储器的**逻辑容量由主存和外存容量之和以及CPU可寻址的范围来决定**，其运行速度接近于主存速度，成本也下降。

**虚拟存储器是一种性能非常优越的存储器管理技术。**



**实现方式主要有以下三种：**

**1.请求分页系统**

该系统是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的页式虚拟存储系统。它允许只装入若干页的用户程序和数据（而非全部程序）就可以启动运行，以后再通过调页功能和页面置换功能陆续把将要使用的页面调入主存，同时把暂不运行的页面置换到外存上，置换时以页面为单位。

**2.请求分段系统**

该系统是在分段系统的基础上增加了请求调段和分段置换功能所形成的段式虚拟存储系统。它允许只装入若干段的用户程序和数据就可以启动运行，以后再通过调段功能和置换功能将不运行的段调出，同时调入将要运行的段，注意：置换时以段为单位。

**3.请求段页式系统**

该系统是在段页式系统的基础上增加了请求调页和页面置换功能所形成的段页式虚拟存储系统。

#### 4请求分页管理的实现

**请求分页：**在纯分页系统的基础上增加了**请求调页功能**、**页面置换功能**所形成的页式虚拟存储系统，它是目前常用的一种虚拟存储器的方式。

**页表机制：**在纯分页的页表机制上形成的，由于只将应用程序的一部分调入主存，还有一部分仍在磁盘上，故需在页表中再增加若干项（如状态位、访问字段和辅存地址等）供程序（数据）在换进、换出时参考。

**地址变换机构：**在分页系统的地址变换结构的基础上增加了某些功能，如产生和处理缺页中断、从主存中换出一页实现虚拟存储。

**缺页中断：**在请求分页系统中，每当所要访问的页面不在主存时便要产生一个缺页中断，请求OS将所缺的页调入主存，这是由缺页中断机构完成的。

**缺页中断与一般中断的主要区别如下：**

1. 缺页中断在指令执行期间产生和处理中断信号，而一般中断在一条指令执行完，下一条指令开始执行前检查和处理中断信号。
2. 发生缺页中断时，返回到被中断指令的开始重新执行该指令，而一般中断返回到下一条指令开始。
3. 一条指令在执行期间可能会产生多次缺页中断。

#### 5页面置换算法

对于页面虚拟存储系统，在进程运行过程中，如果发生缺页，此时主存中又无空闲块时，为了保证进程能正常运行，必须从主存中调出一页程序（或数据）送磁盘的对换区。但将哪个页面调出，需要根据页面置换算法确定。

置换算法的好坏将直接影响系统的性能，不适当的算法可能会导致系统发生抖动。即刚被换出的页很快又被访问，需重新调入，导致系统频繁地更换页面，以至于一个进程在运行中把大部分时间花费在完成页面置换的工作上，这种现象称为系统发生了抖动(颠簸)。请求分页系统的核心问颗是选择合适的页面置换算法。

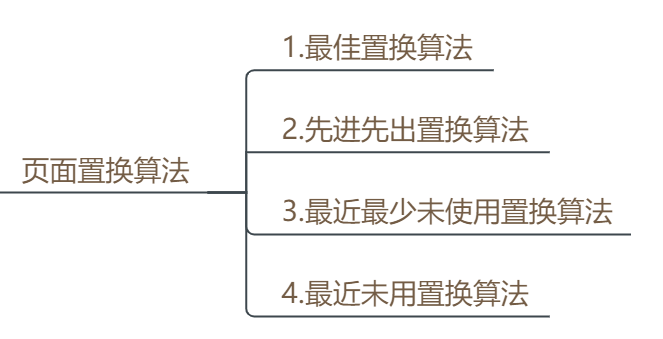


图 16页常用的面置换算法

1. **最佳置换算法**

一种理想化的算法。选择那些永不使用的，或者是在最长时间内不再被访问的页面置换出去。

这种方法性能最好，但实际上难于实现。并且要确定哪一个页面是未来最长时间内不再被访问的是很难的，所以该算法通常用来评价其他算法。

1. **先进先出算法**

一种最直观、性能最差的算法。该算法总是淘汰最先进入主存的页面，即选择在主存中驻留时间最久的页面予以淘汰。它有Belady异常现象。

Belady现象，是指如果对一个进程未分配它所要求的全部页面，有时就会出现分配的页面数增多但缺页率反而提高的异常现象。例如，对于页面访问序列“1，2，3，4，1，2，5，1，2，3，4，5"，当分配的物理块从3块增加到4块时，有缺页次数增加、缺页率提高的异常现象。

1. **最近最少未使用算法**

该算法是选择最近最少未使用的页面予以淘汰，系统在每个页面设置一个访问字段，用于记录这个页面自上次被访问以来所经历的时间T，当要淘汰一个页面时，选择T最大的页面，

1. **最近未用置换算法**

该算法将最近一段时间未引用过的页面换出。该算法为每个页面设置一位访问位（0或1），将主存中的所有页面都通过链接指针链成一个循环队列。当某页被访问时，其访问位置1。在选择一页淘汰时，检查其访问位，如果是0，则选择该页换出;若为1，则重新置为0，暂不换出该页，在循环队列中检查下一个页面，直到访问位为0的页面为止。

#### 6工作集

**1）由来**

程序在运行中所产生的缺页情况会影响程序的运行速度及系统性能，而缺页率的高低又与每个进程所占用的物理块数目有关。

那么，究竟应该为每个进程分配多少个物理块才能把缺页率保持在一个合理的水平上?否则会因为进程频繁地从辅存请求页面而出现抖动现象。为了解决这一问题，引入了工作集理论。**2）介绍**

（1）一个进程当前正在使用的逻辑页面集合，可表示为二元函数 W(t,△)。

（2）W(t,△)

* t 是当前的执行时刻
* △ 称为工作集窗口，即一个定长的页面访问时间窗口
* W(t,△) 是指在当前时刻 t 前的 △ 时间窗口中的所有访问页面所组成的集合
* |W(t,△)| 指工作集的大小，即页面数目