# 连续存储分配方式

## 单一连续分配

- 最简单的一种存储管理方式，只能用于单用户、单任务的操作系统中。

- 存储管理方法：将内存分为系统区和用户区。作业一旦进入内存，就要等待它运行结束后才能释放内存。

## 固定分区分配

- 分区分配方式是满足多道程序设计需要的一种最简单的存储管理方法。

- 存储管理方法：将内存空间划分为若干个固定大小的分区，除OS占一区外，其余的每一个分区装入一道程序。分区的大小可以相等，也可以不等，但事先必须确定，在运行时不能改变。即分区大小及边界在运行时不能改变。

- 分区说明表：以记录分区号、分区大小、分区的起始地址及状态（已分配或未分配）。

## 动态分区分配

- 动态分区分配又称为可变式分区分配，是一种动态划分存储器的分区方法。

- 存储管理方法：不事先对内存划分分区，而是在作业进入内存时，根据作业的大小动态地建立分区，并使分区的大小正好适应作业的需要。因此系统中分区的大小和数目是可变的。

- 特点：管理简单，需小量的软件和硬件支持。分区大小与进程的大小相等，从而主存的利用率有所提高。

- \*\*1.分区分配中使用的数据结构。\*\*

1.空闲分区表：用来登记系统中的空闲分区(分区号,分区起始地址,分区大小及状态)。

2.空闲分区链：将系统中的空闲分区链接起来，构成空闲分区链。每个空闲分区的起始部分存放相应的控制信息(如大小,指向下一空闲分区的指针等)。

- \*\*为了将一个作业装入内存，按照一定的分配算法从空闲分区中选 出一个满足作业需求的分区分配给作业，同时修改空闲分区表（链）中相应的信息。\*\*

\*\*(1) 首次适应算法\*\*

- 空闲分区（链）按地址递增的次序排列。在进行内存分配时,从空闲分区表/链首开始顺序查找,直到找到第一个满足其大小要求的空闲分区为止。

- 按照作业大小，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分区表（链）中。

- 特点：优先利用内存低地址部分的空闲分区,从而保留了高地址部分的大空闲区。但由于低地址部分不断被划分,致使低地址端留下许多难以利用的很小的空闲分区(碎片或零头),而每次查找又都是从低地址部分开始, 增加了查找可用空闲分区的开销。

(2) \*\*循环首次适应算法\*\*

- 由首次适应算法演变而来。在为作业分配内存空间时,不再每次从空闲分区表/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,直到找到第一个能满足其大小要求的空闲分区为止。

- 按照作业大小，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分区表/链中。

- 特点： 使存储空间的利用更加均衡，不致使小的空闲区集中在存储区的一端，但这会导致缺乏大的空闲分区。

(3) \*\*最佳适应算法\*\*

- 空闲分区表/链按容量大小递增的次序排列。在进行内存分配时，从空闲分区表/链的首开始顺序查找，直到找到第一个满足其大小要求的空闲分区为止。

- 即把既满足作业要求又与作业大小最接近的空闲分区分配给作业。

- 特点：若存在与作业大小一致的空闲分区,则它必然被选中；若不存在与作业大小一致的空闲分区，则只划分比作业稍大的空闲分区，从而保留了大的空闲分区；往往使剩下的空闲区非常小,从而在存储器中留下许多难以利用的小空闲区（碎片或零头）。

(4) 最坏适应算法

- 空闲分区表/链按容量大小递减的次序排列。在进行内存分配时，从空闲分区表/链的首开始顺序查找，直到找到第一个比作业大的空闲分区为止。

- 特点：总是挑选满足作业要求的最大的分区分配给作业。这样使分给作业后剩下的空闲分区也较大，可装下其它作业。但由于最大的空闲分区总是因首先分配而划分，当有大作业到来时，其存储空间的申请往往会得不到满足。

## 可重定位分区分配

1、碎片问题

- 由于内存的不断分配和回收，内存逐渐被分割成一些零散的空闲存储区，这种内存中无法被利用的存储空间称为“零头”或“碎片”。

- 内部碎片：指分配给作业的存储空间中未被利用的部分。如固定分区中存在的碎片。

- 外部碎片：指系统中无法利用的小的空闲分区。如动态分区中存在的碎片。

2.碎片问题的解决方法——拼接/紧凑/紧缩技术

3.可重定位分区分配方式主要特点：可以充分利用存储区中的“零头/碎片”，提高主存的利用率。 但若 “零头/碎片”太多，则拼接频率过高会使系统开销加大。

# 离散存储分配方式

## 基本分页存储分配

### 1.页和块

①内存空间划分成若干固定大小的块，各块从0开始顺序编号

②程序划分成若干个与块大小相同的页,各页从0开始顺序编号。

③以块为单位,将程序的若干页装入到多个可能不相邻的块中。

### 2.页表（页面映像表）

① 记录了页面在内存中对应的块号；

②页表一般存放在内存中；

③页表的基址及长度由页表寄存器给出；

④访问一个字节的数据/指令需访问内存2次(页表一次,内存一次),所以出现内存访问速度降低的问题。

### 3.逻辑地址结构

### 4.地址变换机构

### 5.快表（联想寄存器）

(1)一种特殊高速缓冲寄存器。

(2)其中保存当前进程页表的一部分或全部。

(3)首先在快表中寻找逻辑地址的页号，若命中，就找出其对应的物理块；若未命中，再到页表中查找，并将该表项复制到快表。

(4)若快表内容满，则按某种算法淘汰某些页

### 6.多级页表

### 7.分页系统的优点和不足

优点：显著提高了内存利用率。

不足： “页”是一个物理单位，不具有完整的逻辑意义。不能满足用户和程序员在编程和使用上的多方面需求。

## 基本分段存储分配

### 分段存储管理方式的引入

- 方便编程：程序通常按逻辑关系分为若干个段，每个段从0编址，并有名字和长度，访问的逻辑地址由段名和段内偏移量决定。

- 信息共享：共享是以信息为逻辑单位。页是存储信息的物理单位，段是信息的逻辑单位。

- 信息保护：保护也是以信息为逻辑单位。

- 动态增长：实际应用中，某些段（数据段）会不断增长，其它存储管理方法均难以实现。

- 动态链接：动态链接以段为单位。

## 段页式存储分配

### 系统中设段表和页表,均存放于内存中，读一字节的指令或数据须访问内存三次。