8. 内存管理

逻辑地址与物理地址:

逻辑地址是操作系统为程序提供的地址，方便程序访问变量。逻辑地址由**段地址和偏移地址**组成。

物理地址是内存中的真实地址，是 CPU 通过地址总线访问内存单元的地址。物理地址和逻辑地址的转换是由内存管理单元（MMU）完成的。

地址重定位（地址映射）:

将逻辑地址（虚地址）映射到实际的物理地址（实地址）的地址转换功能叫做**地址映射**，也叫**地址重定位**。

1. 静态重定位（装入时进行），无需硬件支持，程序装入内存的过程中完成:

2. 动态重定位（边执行边修改），需要硬件支持（重定位寄存器）:

总结:

静态地址重定位：即在程序装入内存的过程中完成，是指在程序开始运行前，程序中的各个地址有关的项均已完成重定位，地址变换通常是在装入时一次完成的，以后不再改变，故成为静态重定位。

优点：无需硬件支持
缺点：1）程序重定位之后就不能在内存中搬动了；2）要求程序的存储空间是连续的，不能把程序放在若干个不连续的区域中。

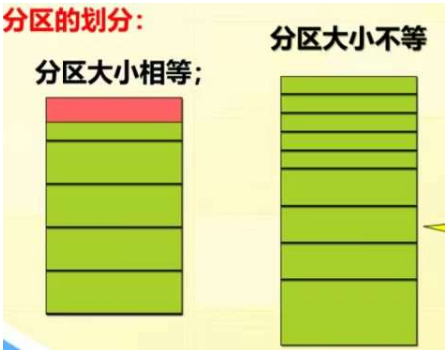
动态地址重定位：不是在程序执行之前而是在程序执行过程中进行地址重定位。更确切的说，是在每次访问内存单元前才进行地址变换。动态重定位可使装配模块不加任何修改而装入内存，但是它需要硬件一定位寄存器的支持。
优点：1）目标模块装入内存时无需任何修改，因而装入之后再搬迁也不会影响其正确执行，这对于存储器紧缩、解决碎片问题是极其有利的；2）一个程序由若干个相对独立的目标模块组成时，每个目标模块各装入一个存储区域，这些存储区域可以不是顺序相邻的，只要各个模块有自己对应的定位寄存器就行。
缺点：需要硬件支持。

连续存储管理方式（分配和回收）：

一、固定分区方式：

概念：将内存空间划分为若干个固定大小的区域，每个分区中只装入一道作业。

划分：



（直接标记分区是否空闲 来分配和回收）

注意：会产生**内部碎片**！降低内存利用率。

二、可变分区方式：

1. 分配算法

a. 首次适应算法

概念：该算法从空闲分区链首开始查找，直至找到一个能满足其大小要求的空闲分区为止。然后再按照作业的大小，从该分区中划出一块内存分配给请求者，余下的空闲分区仍留在空闲分区链中。

特点：该算法倾向于使用内存中低地址部分的空闲区，在高地址部分的空闲区很少被利用，从而保留了高地址部分的大空闲区。显然为以后到达的大作业分配大的内存空间创造了条件。

缺点：低地址部分不断被划分，留下许多难以利用、很小的空闲区（**外部碎片**），而每次查找又都从低地址部分开始，会增加查找的开销。

b. 最佳适应算法

概念：该算法总是把既能满足要求，又是最小的空闲分区分配给作业。所有的空闲区按其大小排序后，以递增顺序形成一个空白链（**由小到大**）。

特点：每次分配给文件的都是最合适该文件大小的分区。

缺点：内存中留下许多难以利用的小的空闲区（**外部碎片**）。

c. 最坏适应算法

概念：最坏适应算法是将输入的作业放置到主存中与它所需大小差距最大

的空闲区中。空闲区大小**由大到小**排序。

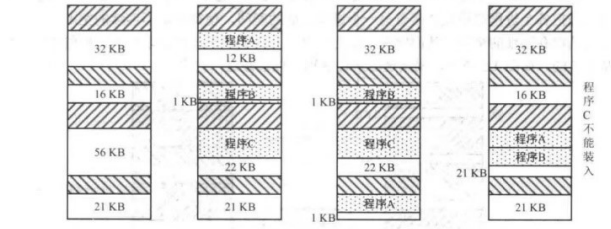
特点：尽可能地利用存储器中大的空闲区。

缺点：绝大多数时候都会造成资源的严重浪费甚至是完全无法实现分配。

三个算法图例

例题 4-2：某系统内存空间分区分配状态如图 4-3(a)所示，程序 A 运行请求 20 KB，程序 B 运行请求 15 KB，程序 C 运行请求 34 KB，分别采用首次适应算法、最佳适应算法和最坏适应算法讨论内存分配情况。

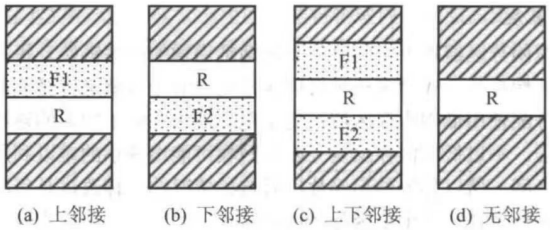
解答：分别采用三种算法的分配情况如图 4-3(b)、(c)、(d)所示。



(a) 系统内存空间分区分配状态 (b) 首次适应算法 (c) 最佳适应算法 (d) 最坏适应算法

图 4-3 空间分配状态及三种算法分配情况

2. 回收算法



(a) 上邻接 (b) 下邻接 (c) 上下邻接 (d) 无邻接

图 4-4 内存回收的四种情况

(1) 上邻接：指的是待回收分区 R 在低端地址与另一个空闲分区 F1 相邻。回收 R 时，R 与 F1 合并成为新的 F1 空闲分区，分区的起始地址不变，但是大小变为 F1 与 R 大小之和。

(2) 下邻接：指的是待回收分区 R 在高端地址与另一个空闲分区 F2 相邻。回收 R 时，R 与 F2 合并成为新的 F2 空闲分区，分区的起始地址变为原来 R 的起始地址，并且大小变为 F2 与 R 大小之和。

(3) 上下邻接：指的是待回收分区 R 在低端地址与空闲分区 F1 相邻，同时在高端地址与空闲分区 F2 相邻。回收 R 时，R、F1 和 F2 合并成为新的 F1 空闲分区，分区的起始地址不变，仍为原来 F1 的起始地址，但是大小变为 R、F1 和 F2 大小之和。并且原来的 F2 分区节点要从空闲分区链中删除，导致空闲分区数量减 1。

(4) 无邻接：指的是待回收分区 R 不与任何其他空闲分区相邻。回收 R 时，需要在合适的位置创建新的空闲分区节点，这样导致空闲分区数量加 1。

3. 伙伴系统

- 整个可分配的分区大小2^U
- 需要的分区大小为$2^{U-1} < s \le 2^U$时，把整个块分配给该进程；
  - 如$s \le 2^{i-1}$，将大小为2^i的当前空闲分区划分成两个大小为2^{i-1}的空闲分区
  - 重复划分过程，直到$2^{i-1} < s \le 2^i$，并把一个空闲分区分配给该进程

数据结构：

空闲块按大小和起始地址组织成二维数组

初始状态：只有一个大小为 2^u 的空闲块

分配过程：

由小到大在空闲块数组中找最小的可用空闲块

如空闲块过大，对可用空闲块进行二等分，直到得到合适的可用空闲块

释放过程：

把释放的块放入空闲数组

合并满足条件的空闲块

合并过程：

大小相同 2^i

地址相邻

低地址空闲块起始地址为 2^(i+1)的位数

特点：空间利用率有所降低，拆分合并开销很大；时间上性能更佳