内存管理

固定分区

- **等分区 or 不等分区**
- 每个分区装一个不大于分区的进程
- 无可用分区时，可以将进程挂起（换入换出）
- 对于大于分区的进程，可以利用“覆盖”
- 缺点：内存利用率低，**内部碎片（小于分区的进程存入分区会产生碎片）**
- 放置算法
  - 等分区：任意选择一个可用分区放入进程
  - 不等分区
    - 将不同大小的分区组成多个队列，不同大小的进程从对应大小的队列中选择分区进行存放
    - 将所有分区组成一个队列，不同大小的进程从中选择一个最适合大小的分区进行存放
      - 分区可以偏小（相比于进程大小）---》覆盖
      - 分区可以偏大（相比于进程大小）---》产生内存碎片

动态分区

- 分区的大小和数量可变
- 进程大小等于分区大小
- 缺点
  - **外部碎片：还没有被分配出去（不属于任何进程），但由于太小了无法分配给申请内存空间的新进程的内存空闲区域。**
    - 压缩技术：对分区进行重新排列，需要采用动态重定位技术，浪费处理器时间
- 放置算法
  - 最佳匹配：搜索和进程大小最接近的内存区域作为分区
    - 复杂度高
    - 会产生较多外部碎片
  - 首次匹配：从内存空间的零地址开始，找到第一个可以放下进程的空间作为分区
  - 临近匹配：不是每次都从首地址查找，而是从上次查找结束的地方开始找，找到第一个可以放下进程的空间作为分区
  - 伙伴系统：进程的大小为S，分配2^k大小的块给进程，其中$2^{k-1} < S < 2^k$
    - 动态分区（分区大小不固定）结合固定分区（分区大小是2^k）

简单分页

- 将内存分为固定大小的**帧**，将进程分为固定大小的**页**，**页的大小等于帧的大小，一般为**2^n
- OS为每个进程维护一个页表
- 有效减少内部碎片：帧和页的大小都很小
- 有效减少外部碎片：采用类似于固定分区的方式，没有外部碎片
-
- 分页的逻辑地址
    - **[页号][偏移量]**
    - 偏移量指在页内的偏移量
    - 拿到页号，通过页表找到帧的起始**物理地址**，加上偏移量得到最终的物理地址
    - *需要判断偏移量小于帧的大小*

简单分段

- 内存按某种方式化为大小不等的段，类似于动态分区
- 分段的逻辑地址
    - **[段号][偏移地址]**
    - 段表还需要维护段的大小
    - 需要判断偏移量小于段的大小

重定位

- 就是把程序的逻辑地址变换成内存中的实际物理地址的过程。
- 在程序的执行过程中进行