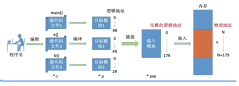
第三章-内存管理

内存管理功能

- 程序的链接与装入
  - 编译
  - 链接
    - 静态链接:运行前链接成一个完整的装入模块
    - 装入时动态链接:装入时采用边装入边链接的方式，便于修改和更新共享
    - 运行时动态链接：执行需要目标模块时才进行链接
  - 装入
    - 绝对装入：程序中的逻辑地址与物理地址完全相同，不需对程序和数据的地址进行修改
    - 可重定位装入:在装入时基于始址对目标程序的相对地址进行修改，要求一次全部装入，且装入后在不能在内存中移动
    - 动态运行时装入：装入程序在装入时不会修改相对地址，等到执行前再借助重定位寄存器修改。可以分配不连续的存储区，可以根据需要动态申请内存
  - 
- 逻辑地址与物理地址：内存机制对应用程序员完全透明，只能使用逻辑地址，操作系统通过内存管理部件MMU根据页表将逻辑地址转换成物理地址
- 进程的内存映像
  - 堆：存放动态分配的变量 栈：用来实现函数调用
- 内存保护：防止内存访问其他的地址空间
  - 上下限寄存器：CPU中设置一对寄存器，用于存放进程在主存中的上下限地址，CPU访问地址时进行比较判断
  - 重定位寄存器和界地址寄存器：界地址寄存器存放最大的逻辑地址，用于比较判断，再和存放起始物理地址的**重定位寄存器**进行相加得到物理地址，只有特权指令可以访问
- 内存共享：对于多用户可共享文件，只需建立多个页表项指向相同的物理地址即可实现无需存储多份相同文件，段式存储中只需一个段表项即可完成共享

连续分配管理方式

- 单一连续分配：内存中只有一个用户程序
- 固定分区分配：将用户空间划分成若干个固定大小的分区一对一分配给每个进程，维护一张分区使用表。碎片多，无法完成内存共享
- 动态分区分配
  - 设置一张空闲分区链（特定方式排序），分配时遍历找到合适的大小的空间用于分配
  - 动态分区分配的内存回收
    - 回收区前/后/前后都有相邻的空闲分区要合并成一个大的空闲区，若没有则建立新的空闲表项插入空闲分区链
- 顺序搜索的内存分配算法
  - 首次适应算法：从头开始找到第一个满足大小的分区，综合性能最好
  - 邻近适应算法：从上次查找结束的位置继续查找，内存高低部分的空闲区以同等概率被分配
  - 最佳适应算法：空闲分区以容量递增排序，找到第一个满足大小的空闲分区，回收空闲分区时涉及重新排序开销大
  - 最坏适应算法：空闲分区以容量递减排序，找到第一个满足大小的空闲分区，回收空闲分区时涉及重新排序开销大
- 基于索引搜索的分配算法
  - 在大型系统中采用索引分配算法：根据大小对空闲分区分类，对于每类空闲分区单独设置一个空闲分区链，设置一个索引表来管理这些链的头指针
  - 快速适应算法：1.根据带下找到能容纳的最小的分区链2.取链的第一块分配
  - 伙伴系统：规定所有分区大小为2^k,并建立k个分区链，分配大小为n的空间时找第i条空闲链（n<2^i），若未找到则找i+1，并将i+1中的分区分成两份(伙伴系统)，一份用于分配，一份加到i的空闲链中
  - 哈希算法：根据空闲分布规律建立哈希函数，构建以空闲分区大小为关键字的哈希表，每个表项记录一个对应的空闲分区链头指针，分配时根据大小计算哈希表位置

非连续分配管理方式

- 基本分页存储系统
  - 地址变换机构
    - 页号是隐含的，页表中无需存储
    - 整个过程都是由硬件完成
    - 设置一个页表寄存器PTR用于存放页表始址和页表长度，页表始址用于加页号得到对应表项的物理地址，长度用于判断是否越界
  - 页表地址：页号+页内偏移量
    页表项：（页号）物理块号
  - 快表
  - 两级页表：地址:一级页号(页目录号，二级页表的始址)+二级页号+页内偏移
    外层页表的表项中存放的是二级页表分页的始址，增加了访问主存的次数
- 段页式存储管理
  地址：段号+页号+页内偏移量
  段表项：页表长度（用于判定地址合法）+页表始址
  页表项：物理块号
- 基本分段存储系统
  - 系统将进程划分逻辑段，段号和段内偏移量必须由用户显式给出（高级语言由编译程序给出）
  - 段地址：段号+段内偏移量
    段表项：（段号隐式给出）段长+本段始址
    物理地址=（本段始址+段内偏移）加法得出