9.10内存管理基础、虚拟内存

2153401 赵一婷

- ch9 内存管理基础
  - 引言
    - 程序执行前
      - 编译：编译程序将用户代码（--->地址形式：符号）编译成若干个目标模块（--->地址形式：可重定位的地址）
      - 链接：（链接程序）将目标模块和所需库函数链接在一起，形成完整的装入模块
      - 装入：（装入程序）将装入模块装入内存 ---->内存映像
    - 程序链接技术
      - 常用链接方法
        - 静态链接
        - 装入时动态链接：在装入内存时，边装入边链接
        - 运行时动态链接：在程序运行过程中，当需要该目标模块时才链接（效率高！
    - 程序装入技术（操作系统关心
      - 过程：名空间---编译链接---->地址空间（应用程序空间/逻辑空间/虚拟地址空间/相对空间：应用程序地址/逻辑地址/相对地址） ----装入----> 存储空间（真实空间/物理空间：物理地址/绝对地址/存储空间地址）
      - 常用技术
        - 地址再定位技术：将逻辑地址向物理地址映射，由操作系统中的装入程序来完成
        - 常用装入技术
          - 绝对装入技术（固定地址再定位）：在编译链接时直接制定程序在执行时访问的实际存储地址，程序地址空间和内存地址空间是一一对应的
            - 优点：装入过程简单
            - 缺点：过于依赖硬件结构，不适于多道程序系统
          - 可重定位装入技术：可执行文件中列出各个需要重定位的地址单元和相对地址值，装入时再根据所定位的内存地址去修改每个重定位地址项，添加相应偏移量
            - 两种地址再定位方式
              - 静态再定位：装入程序在程序执行之前进行地址再定位，一旦地址定位完成后，程序执行期间不会发生变化。
                - 优点：易于实现，无需硬件支持
                - 缺点：

        - 程序再定位后不能移动，不利于内存的有效分配
        - 程序在存储空间中只能连续分配
    - 动态再定位：程序在装入内存时，不修改逻辑地址，在访问物理内存之前，再实时地将逻辑地址转换成物理地址。
        - 优点
            - 程序在执行过程中可以移动，有利于内存充分利用。
            - 程序不必连续存放在内存中，可分散在内存若干个不同区域，只需增加几对基址一限长寄存器，每对寄存器对应一个区域。
        - 缺点：需要附加硬件支持，实现存储管理的软件算法比较复杂

- 存储管理思想*****
    - 存储组织结构：寄存器--高速缓存---主存---外存
    - 存储管理目的：
        - 充分利用内存
        - 方便用户使用
        - 解决程序空间比实际内存空间大的问题
        - 存储保护与安全
        - 共享与通信
        - 实现的性能和代价
    - 存储管理任务
        - 存储分配和回收
        - 存储共享
        - 存储保护
        - 存储器扩充
    - 存储管理方案
        - **连续内存分配方法****
            - 单一连续存储管理：内存中只放一个应用程序且连续放 缺点:空间利用率低
            - 分区存储管理
                - 基本思想：把内存分为大小相等或不等的分区，每个进程占用一个或几个分区；操作系统占用其中一个分区。
                    - 特点：
                        - 适用于多道程序系统和分时系统
                        - 支持多个程序并发执行
                    - 问题

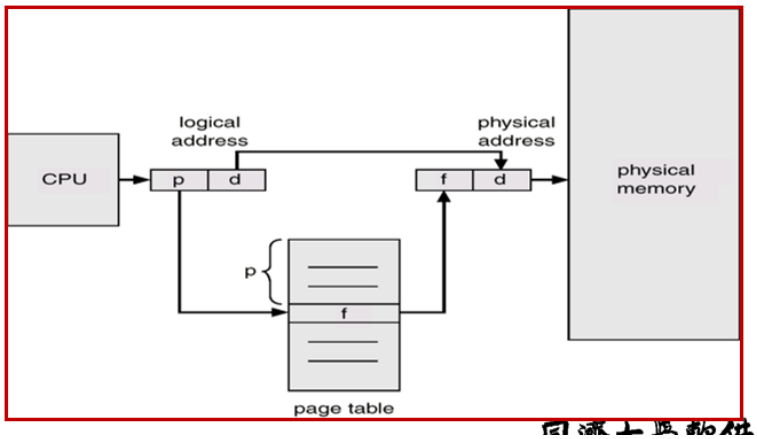
        - 可能存在内碎片和外碎片
        - 难以进行内存分区的共享
- 数据结构：分区表（起始地址，长度，标志)(是全局表，由操作系统维护）
- **固定分区存储管理**：内存划分为若干个固定大小的连续分区
    - 优点
        - 内存利用率提高
        - 可以支持多道程序
        - 实现简单
    - 缺点
        - 程序必须预先能够估计要占用多大的内存空间
        - 内碎片造成浪费
        - 分区总数固定，限制了并发执行的程序数目
- **动态分区存储管理**：动态创建分区
    - 数据结构：分区表
    - 优点：没有内碎片
    - 缺点：存在外碎片
- 常用分区管理算法***
    - **最先适配算法**：分区链表按照地址从小到大排序
        - 实质：尽可能利用存储区低地址空闲区，尽量在高地址部分保存较大空闲区，以便一旦有分配大空闲区要求时，容易得到满足
        - 算法优点：分配简单，合并相邻空闲区也比较容易
        - 算法缺点：前面空闲区往往被分割的很小，查找次数较多。
    - **循环最先适配算法**：按分区先后次序，从上次分配的分区起查找（到最后分区时再回到开头），找到符合要求的第一个分区
        - 优点：算法的分配和释放的时间性能较好，使空闲分区分布得更均匀
        - 缺点：但较大的空闲分区不易保留。
    - **最佳适配算法**：空闲区间由小到大排序
        - 优点 ： 较大的空闲分区可以被保留。
        - 缺点：空闲区是按大小而不是按地址顺序排列的，因此　释放时，要在整个链表上搜索地址相邻的空闲区，合并后，又要插入到合适的位置。（便于分配但不便于合并）
    - **最坏适配算法**：空闲区间由大到小排序
        - 优点 ： 分配时，只需查找一次就可成功，分配算法很快。
        - 缺点：最后剩余分区会越来越小，无法运行大程序

- 分区算法存在的问题
  - 碎片问题 ---->解决方法：紧凑技术----->离散分配方法
- 分区保护问题（了解
  - 界限寄存器：定位寄存器和界限寄存器
  - 保护键
- 内存扩充技术：借助大容量辅存在逻辑上实现内存扩充，以解决内存容量不足的问题。
  - 原因/背景：大程序、小空间
  - 常用方法
    - 覆盖技术
      - 覆盖技术缺点
        - 编程时必须划分程序模块和确定程序模块之间的覆盖关系，增加编程复杂度。
        - 从外存装入覆盖文件，以时间延长换取空间节省
    - 交换技术
      - 整体交换：交换以整个进程为单位，也称为进程交换
        - 目的：解决内存紧张，进一步提高内存利用率
      - 部分交换：以分页、分段交换为基础，也称为页面交换，分段交换；（内存与磁盘间的交换
        - 目的：支持虚拟存储系统
      - 优点：
        - 增加并发运行的程序数目（每个进程都放一部分
        - 给用户提供适当的响应时间
        - 编写程序时不影响程序结构（不需要程序员来做、OS做
      - 缺点：换入和换出的控制增加处理机开销。（换入换出的频率用“抖动”来衡量
    - 覆盖技术与交换技术比较

- **离散内存分配方法**
  - 分页存储管理（分配单位是页）*****
    - 管理思路：
      - 划分用户空间：由系统自动完成，把用户程序按逻辑页划分成大小相等的部分，称为页（虚页）
      - 划分内存空间：把用户程序按逻辑页划分成大小相等的部分，称为页（虚页）
      - 分配内存：以页为单位进行分配，并按任务页数多少来分配。逻辑上相邻的页，物理上不一定相邻。

- 数据结构：
  - 进程页表：存放逻辑页号和具体内存块号相应的关系；
    - 放在内存中，属于进程的现场信息
    - 每一个进程都有一个页表
  - 物理页面表：描述物理内存空间的分配使用状况。（系统维护
  - 请求表：描述各个进程页表位置和大小，也可结合到各进程PCB里。（系统维护
- 内存分配过程：1.计算所需块数N 2.查看位示图看是否还有N个空闲页面块 3.如有，则页表长度设为N，可填入PCB中 4.申请页表区，把页表起始地址填入PCB 5.依次分配N个空闲块，将块号和页号填入页表，修改位示图
- 硬件支持
  - 页表始址寄存器
  - 页表长度寄存器
  - 联想寄存器——快表（在内存中）：可以缩短查找时间，实现按内容查找，即逻辑页号 - >物理页号（TLB
  - CPU中取出指令，指令地址为p+d，在页表中取出对应的物理页号f，与偏移量d一同构成真实物理地址，在内存中查找

    地址映射

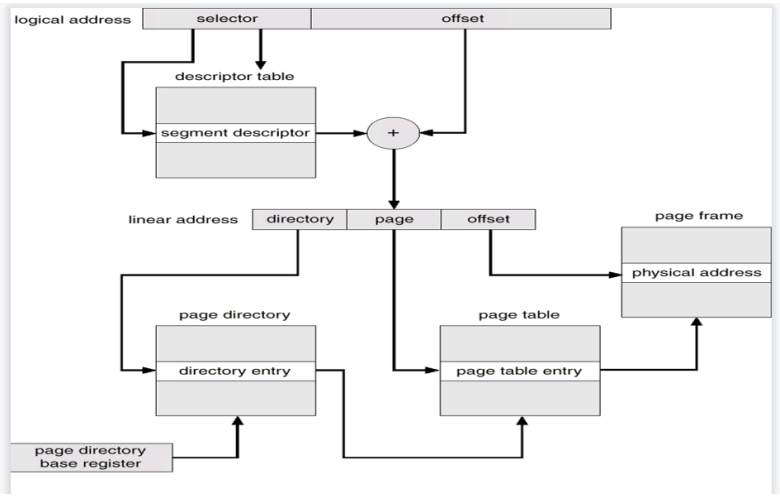
    

- **注意点！：操作系统负责提供页表信息，CPU用于拼接地址与偏移量（计算真实物理地址），所以CPU内又多了一个映射+拼接的控制器MMU**
- 优点
  - 解决了碎片问题（但还是有可能产生外碎片）
  - 便于管理
- 缺点
  - 不易于实现程序共享（并且信息会被4k4k割裂开）
  - 不便于动态链接
- 两级页表

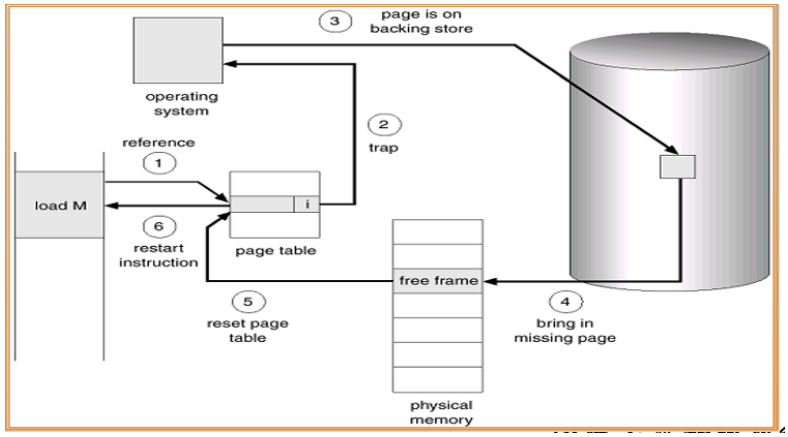
        - 目的：为了减少进程页表所占有的空间（不同时生成全部的页表，而是用哪部分生成哪部分
        - 一级页表叫页目录，目录是一个物理的页，每个目录项对应的页表也是一页
        - 优缺点：牺牲速度换取空间
- 段式存储管理（分配单位是段）***
    - 主要动力是：满足用户需求与提高内存利用率
    - 管理思想
        - 划分用户空间：
            - 按程序自身的逻辑关系划分为若干个程序段
            - 每个程序段都有一个段名，且有一个段号
            - 段号从0开始，每一段从0开始编址，段内地址是连续的
        - 划分内存空间
            - 内存空间被动态的划分为若干个长度不相同的区域，这些区域被称为物理段;
            - 每个物理段由起始地址和长度确定。
        - 分配内存
            - 以段为单位分配内存，每一个段在内存中占据连续空间;
            - 各段之间可以不连续存放
    - 数据结构
        - 进程段表：记录了段号、段的首（地）址、段长度
            - 每一个进程设置一个段表，放在内存，属于进程现场信息
        - 系统段表：系统内所有占用段
        - 空闲段表：记录空闲段起始地址和长度，可以结合到系统段表中。
    - 硬件支持
        - 段表始址寄存器 保存正在运行进程的段表始址。
        - 段表长度寄存器 保存正在运行进程的段表长度。
        - 联想存储器
            - 保存正在运行进程的段表的子集;
            - 快表项目：段号、段始址、段长度、标识（状态）位、访问位（淘汰位）。
    - 优点
        - 便于动态申请内存
        - 管理和使用统一化
        - 便于共享
        - 便于动态链接

- 缺点
  - 产生外碎片
- 段页式存储管理
  - 基本思想
    - 将每一个进程划分为多个段，每一段按照页式的方式再划分
    - 内存空间：页式管理，以页为单位进行分配
    -

  - 数据结构
    - 段表
    - 页表
    - 空闲块管理同页式管理(因为是按页分配的
    - 内存分配同页式管理
  - 硬件支持
    - 段表始址寄存器　段表长度寄存器　相联存储器（快表）
    - (要会画)



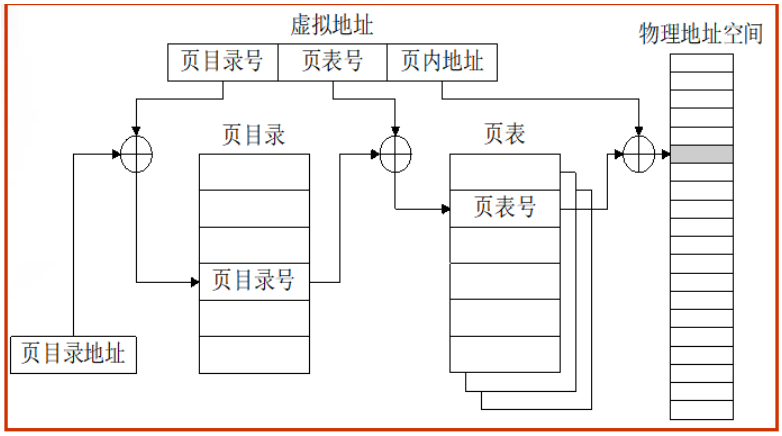
- **虚拟存储器——带有中断机制的页式管理机制**
  - 虚拟内存是一个真实的物理存在，其最大容量是由CPU地址结构确定的，实际容量是由内存和硬盘交换区之和确定的
  - 基本思想（要会描述****
    - 程序装入：只需将当前需要执行的部分页或段读入到内存
    - 程序执行中：若待访问数据不在内存（发生缺页或缺段），则发生缺页中断，处理器通知操作系统将相应的页或段调入内存；操作系统可将暂不使用的页或段调出保存在外存上

- 优点：
  - 存放更大的程序
  - 提供更大的用户空间： 提供给用户可用的虚拟内存空间大于物理内存
  - 更多程序并发执行
  - 易于开发：覆盖技术比较，不影响编程时的程序结构。？
- 缺点：牺牲速度换空间利用率，以管理的相对复杂度换取了空间利用率的提升
- 虚拟存储特征
  - 不连续性
    - 物理内存分配的不连续性；
    - 虚拟地址空间使用的不连续性。
  - 部分交换
  - 大空间：提供大范围的虚拟地址空间，其总容量不超过物理内存和外存交换区容量。
- 虚拟存储技术
  - **请求页式管理*****
    - 原理：只有在页面需要时，才将其载入内存
    - 内存分配由两种策略：1.固定分配 2.可变分配 固定分配易导致频繁出现缺页中断
    -
      

    - 优点：需要更少的输入输出、更小的内存、更快的响应、更多的用户
    - 页表结构（页号：逻辑页号、块号：物理页号、外存：在磁盘上的位置、修改位：记录表是否修改，调回时是否要写回磁盘

页号	中断位	内存块号	外存地址	访问位	修改位

      - 为缩短查找时间，多级页表中的每级都可以装入到联想存储器中，并按照cache原理进行更新。
    -



二级页表地址映射

- 页面置换算法****
  - 先进先出算法(FIFO)：选择建立最早的页面被置换。
    - 性能较差，抖动现象
  - 最佳算法（OPT）：选择“未来不再使用”或“在离当前最远位置上出现的”
  - 最近最少使用页面淘汰算法(LRU)
  - 最不常用算法(LFU)：选择到当前时间为止被访问次数最少的页面被置换
  - 轮转算法(clock)
- 描述置换算法的性能：缺页率（抖动的多少、缺页的次数
- 请求段式管理

- 虚拟存储策略***
  - 调入策略
    - 请求调页：只调入发生缺页时所需的页面
      - 优点：易于实现
      - 缺点：对外存I/O次数多，开销较大
    - 预调页： 发生缺页需要调入某页时，一次调入该页以 及相邻的几个页（常发生在程序装入时
      - 优点：提高调页的I/O效率
      - 缺点：基于预测，若调入的页在以后很少被访问，则效率低。
  - 页面调入来源
    - 交换区：进程装入时，将全部页面复制到交换区，以后总是从交换区调入。
      - 调入速度快，要求交换区空间较大。
    - 文件区：未被修改的页面，直接从文件区读入，被置换时不需调出；

        - 已被修改的页面，被置换时需调出到交换区，以后从交换区调入。
    - 调出策略：确定何时将已修改页面调出到外存上
        - 请求调出：页面被置换时才调出
            - 缺点：调入所缺页面之前还要调出已修改页面，缺页进程等待时间较长
        - 预掉出： 页被置换之前就调出，因而可以成批调出多个页面。
            - 缺点：形成不必要的开销
    - 负载控制策略：决定驻留在内存中的进程数目，在避免发生抖动的前提下，尽可能提供进程并发水平

以上内容整理于 幕布文档