第四章 存储器管理

- 存储器的多层结构：CPU寄存器、主存、辅存。
- 程序的装入：
  - 绝对装入。
  - 静态重定位（可重定位装入）：装入时完成地址变换。
  - 动态重定位（动态运行时装入）：程序真正执行时进行地址转换。用**重定位寄存器**实现。
- 程序的链接：
  - 静态链接：装入前链接成完整模块。
  - 装入时动态链接：边装入边链接。
  - 运行时动态链接。
- 连续分配存储管理方式：
  - 单一连续分配：一道程序独占内存。
  - 固定分区分配：分区大小可相等，可不等。
  - 动态分区分配：
    - 数据结构：空闲分区表、空闲分区链。
    - 基于顺序搜索的动态分区分配算法：
      - 首次适应算法(first fit, FF)：从低址部分开始找。
        - 优点：优先利用了低址部分的内存空间，保留了高址部分的大空闲分区。
        - 缺点：低址部分不断划分，产生小碎片；每次查找从低址部分开始，增加了查找的开销。
      - 循环首次适应算法(next fit, NF)：从上次找到的空闲分区的下一个空闲分区开始查找。
        - 优点：空闲分区分布均匀，减少查找开销。
        - 缺点：缺乏大的空闲分区。
      - 最佳适应算法(best fit, BF)：将空闲分区按容量递增排序。
        - 优点：保留大分区，但有很多小的外部碎片。
        - 缺点：产生许多难以利用的小空闲区。
      - 最坏适应算法(worst fit, WF)：将空闲分区按容量以从大到小的顺序形成空闲分区链。
        - 优点：减少小碎片，提高查找效率。
        - 缺点：但缺乏大的空闲分区。
      - 快速适应算法/分类搜索法(quick fit)：将空闲分区按容量大小进行分类，对于每一类具有相同容量的所有空闲分区，单独设立一个空闲分区链表，系统中存在多个空闲分区链表。
        - 优点：查找效率高，仅需根据进程长度，寻找到能容纳它的最小空闲分区链表，取下第一块即可。
        - 缺点：在分区归还主存时算法复杂，系统开销较大。
      - 伙伴系统：linux中使用。
- 对换：将内存中暂时不能运行的进程换到外存上。
  - 对换的类型：

        - 整体对换：以进程为单位
        - 部分对换：以页或段为单位
- 磁盘空间分为文件区和对换区。
    - 对文件区管理的目标：主要是提高文件存储空间的利用率，然后是提高对文件的访问速度。
    - 对对换区管理的目标：提高进程换入换出的速度，然后才是提高文件存储空间的利用率。
- 分页存储管理方式：
    - 页框=页帧=内存块=物理块=物理页面。
    - 页表：系统为每个进程建立，实现了从页号到物理块号的地址映射，页表通常存在PCB(进程控制块)中。
        - CPU取数据时，访问两次内存。
    - 如何进行地址转换：
        - 页号=逻辑地址/页面长度
        - 页内偏移量=逻辑地址%页面长度
    - 快表/联想寄存器(associative memory, AM / translation lookaside buffer, TLB)。
        - 在寄存器中，不在内存中。
    - 多级页表：减少页表的内存空间占用。
    - 反置页表：全局只有一个页表，每个条目包含了一个虚拟地址和指向该页的所属进程信息。
- 分段存储管理方式：
    - 分段存储管理的引入：主要是为了满足用户和程序员的需要。
    - 方便编程、信息共享、信息保护、动态增长、动态链接
- 分段与分页的主要区别：
    - 分页的地址空间是一维的，分段的地址空间是二维的。
    - 页是信息的物理单位。
    - 段是信息的逻辑单位。
- 段页式系统中，获得指令或数据，需要三次访问内存。

第五章 虚拟存储器

- 常规存储器管理方式特征：
    - 一次性
    - 驻留性
    - 局部性原理：Denning于1968年指出
- 虚拟存储器的特征：
    - 多次性：作业中的程序和数据，允许被分成多次调入内存运行。
    - 对换性：作业中的程序和数据，无须在作业运行时常驻内存。
    - 虚拟性：逻辑上扩充内存容量。
- 分页请求系统的硬件支持：
    - 请求分页的页表机制
    - 缺页中断机构

    - 地址变换机构
- 请求分页存储管理方式
    - 请求页表机制：
        - 页号+物理块号+...
        - 状态位/存在位P：指示该页是否已调入内存。
        - 访问字段A：记录本页在一段时间内被访问的次数；或本页最近已有多久时间未被访问。
        - 修改字段M：标识该页在调入内存后是否被修改过。
        - 外存地址
    - 缺页中断机构：指令执行期间产生中断。
    - 地址变换机构（步骤）：
        - 首先检索快表，从中找出所要访问的页，若找到，便修改访问位。对于写指令，还要将修改位改为1。然后利用页表项中给出的物理块号和页内地址形成物理地址。
        - 若快表中没找到该页的页表项，则到内存中找页表，从找到的页表项中的状态位判断该页是否已调入内存。若该页未调入内存，产生缺页中断，请求OS从外存把该页调入内存。
    - 内存分配策略：固定分配局部置换、可变分配全局置换、可变分配局部置换。
- 页面置换算法：
    - 最佳(optimal)置换算法：看未来不会被访问的页面，淘汰掉。
    - 先进先出(FIFO)：只有FIFO算法会产生Belady异常，当为进程分配的物理块数增大时，缺页次数不减反增
    - 最近最久未使用(Least Recently Used, LRU)算法：看过去
    - 最少使用算法(Least Frequently Used, LFU)：记录页面被访问的频率。使用移位寄存器，每次访问某页时，将移位寄存器的最高位置置1，一定时间自动右移一位。
    - Clock置换算法/最近未用算法(NRU, Not Recently Used)：将内存中所有页面链接成一个循环队列，将访问位为0的换出。
    - 改进型Clock算法：访问位A，修改位M，优先换出00，01，10，11。
- 抖动：系统中运行的进程太多，分配给每个进程的物理块太少，进程运行时频繁地发生缺页。
- 工作集：（Denning于1968年提出）过去某段时间内引用的页面的集合。
- 请求分段存储管理方式：
    - 请求段表机制：
        - 段名、段长、段基址、存取方式
        - 访问字段A、修改位M、存在位PA
        - 增补位：本段运行中是否有动态增长
        - 外存地址
    - 共享段表：
        - 共享进程计数count，当count=0时，系统才回收该段所占内存区。
- 存储保护方法：**界地址法、保护键法。**

第六章 输入输出系统

- IO软件的层次结构：
  - 用户层软件：实现与用户交互的接口
  - **设备独立性软件**：实现用户程序与设备驱动器的统一接口
  - **设备驱动程序**：实现OS对设备发出的操作指令
  - **中断处理程序**：保存被中断进程的CPU环境
  - *硬件*
- IO系统接口：
  - 块设备接口：
    - 数据的存取以数据块为单位的设备，如磁盘。
    - 特征是传输速率高，可寻址，可随机读写磁盘中任意块。
  - 流设备接口/字符设备接口：
    - 数据的存取和传输以字符为单位，如键盘，打印机，摄像头。
    - 特征是传输速率较低，不可寻址。
  - 网络通信接口
- IO设备分类：
  - 按使用特性分类：存储设备、IO设备
  - 按传输速率分类：低速设备：键盘、鼠标；中速设备：打印机；高速设备：磁盘。
- 通常，设备并不直接与CPU进行通信，而是与设备控制器通信。
  - 设备控制器的主要功能：控制一个或多个IO设备，实现IO设备和计算机之间的数据交换。
- IO通道：
  - 引入：承担部分原本由CPU处理的IO任务。
  - IO通道能执行IO指令，没有自己的内存，与CPU共享内存。
  - 通道类型：
    - 字节多路通道：数百个子通道按时间片轮转方式共享主通道。
    - 数组选择通道：使用高速设备，独占通道。
    - 数组多路通道：结合上面两者，具有高传输速率的同时，提高通道利用率。
- 中断：外部IO设备中断
- 陷入：CPU内部中断
- 对IO设备的控制方式：
  - 轮询的可编程IO方式/程序直接控制方式：
  - 中断方式：广泛采用，以字节为单位
  - DMA方式：以数据块为单位，仅在传输数据块的开始和结束时，CPU才干预。
  - IO通道方式：减少CPU对一组数据块读写的干预。
- 假脱机系统(Spooling)
  - 提高IO速度

    - 缓和CPU和低速IO设备之间速度不匹配的矛盾
    - 将独占设备改造为共享设备
    - 实现了虚拟设备功能
- 缓冲池：
    - 输入缓冲区：收容输入、提取收入，完毕后缓冲区挂到空缓冲队列。
    - 收容输出、提取输出，完毕后缓冲区挂到空缓冲队列尾部。
- 磁盘访问时间：寻道时间+旋转延迟时间+传输时间
- 磁盘调度算法：
    - 先来先服务(FCFS)
    - 最短寻道时间优先(SSTF)
    - 扫描算法/电梯算法(SCAN)
    - 循环扫描算法(CSCAN)：单向

第七章 文件管理

- 文件系统层次结构：
    - 用户（程序）
    - 文件系统接口
    - 对对象操纵和管理的软件集合
    - 对象及其属性
- 文件的逻辑结构：
    - 按文件是否有结构：
        - 有结构文件
            - 定长记录
            - 变长记录
        - 无结构文件：流式文件，利用读写指针访问字符。
    - 按文件组织方式分类：
        - 顺序文件
            - 串结构：按存入时间先后排序。
            - 顺序结构
        - 索引文件
        - 索引顺序文件：为一组记录中的第一个记录建立一个索引表项。
- 对文件目录管理的要求：
    - 实现按名存取
    - 提高对目录的检索速度
    - 文件共享
    - 允许文件重名
- 查找目录说明过程：/usr/ast/mbox

    - 首先系统读入第一个文件分量名usr，用它与根目录中各目录项文件名顺序比较，找出匹配者，得到匹配项的索引节点号6。
    - 再从6号索引节点中得知usr目录文件在132号盘块中，将该盘块读入内存。
    - 系统再将第二个分量ast读入，用它与放在132号盘块中的第二级目录中各目录项的文件名比较，找到匹配项，得到ast的目录文件在26号索引节点中，再从26号索引节点中得知/usr/ast在496号盘块中，再读入496号盘块。
    - 然后将第三个分量名mbox读入，用它与第三级目录文件中各目录项的文件名比较，最后得到/usr/ast/mbox的索引节点为60，即在60号索引节点中存放了指定文件的物理地址。目录查询操作到此结束。

第八章 磁盘存储器的管理

- 外存的组织方式
    - 连续组织方式：对文件的访问速度最快，可实现直接存取，但不易插入修改。
    - 链接组织方式：
        - 隐式链接：每个目录项都含有指向链接文件第一个盘块和最后一个盘块的指针。只支持顺序访问，不适合随机访问
        - 显示链接：有FAT（文件分配表）（针对整个磁盘而言）
            - FAT表项：{物理块号，下一块}，支持顺序访问和随机访问。
            - 一个磁盘设置一张FAT，开机时，将FAT读入并常驻内存。而文件索引表是一个文件对应一张表。
    - 早期的一些文件系统
        - FAT12,FAT16
        - FAT32:单个文件长度不大于4GB
        - NTFS
    - 索引组织方式（针对单个文件而言）
        - 单级索引组织方式
        - 多级索引组织方式
        - 增量式索引组织方式/混合索引方式
            - UNIX系统采用该种方式，0-9存放直接地址，10-12存放间接地址。
- 文件存储空间管理
    - 空闲表法
    - 空闲链表法
    - 位示图法
    - 成组链接法(UNIX系统采用)
- 廉价磁盘冗余阵列(RAID, Redundant Array of Inexpensive Disk)：由许多小磁盘组成一个大容量的廉价磁盘冗余阵列。
- 文件操作的系统调用
    - 打开文件open()
    - 检测文件是否访问过access()

- 创建新文件create()
- 读文件read()
- 写文件write()
- 关闭文件close()
- 删除文件unlink()
- 设置当前读写位置lseek()
- 链接文件link()
- 创建特殊文件mknod() 包括有名管道、字符设备、块设备
- 创建无名管道pipe()