4.1 4.2 4.3

存储管理的目的：**主存的分配和管理、提高主存储器的利用率、“扩充”主存容量、存储保护**

**程序的装入方式：绝对装入、可重定位装入（将目标程序的逻辑地址转为物理地址，静态重定位）、动态运行时装入（程序真正要执行时才进行地址转化），重定位需要寄存器**

**链接方式：**

**静态链接：将模块和库函数一次性链接成整体，修改相对地址，变换外部调用符号**

**装入时动态链接：边装入边链接，装入时发生调用即装入引起调用目标模块，便于修改共享**

**运行时链接：将某些链接推迟到运行时，只有发生调用时才装入内存**

**地址保护：自动修改地址、分页(操作系统位于0页、作业1页)、界限寄存器**

单用户(单一连续) 存储管理：内存分系统区、用户区，只有一个进程在内存，利用率低，管理简单，采用静态分配和静态重定位、

固定分区：最简单的多道方案、分多个连续区、每区一个程序、分区长可变可不变、可对每个分区设置一个等待队列，也可只设一个等待队列，优缺点自己补

可变分区：系统依据作业需求和内存状况确定是否分配内存、动态分配回收、连续空闲合并

数据结构：空闲表/链、分配表，优点便于动态分配链接，便于共享内存，但产生碎片

最佳BF：从空闲表找出一个最小且大于需求的空间分配给作业，空闲表从小到大排序

最坏FF：从空闲表找出一个最大且大于需求的空间分配给作业，空闲表从大到小排序

首次NF：**按地址从低到高搜索**，只要找到第一个就分配

下次NF：从上一次查询结束的地方开始查询，地址从高到低排序，只要找到就分配

碎片：不断分割空间而产生的小的空闲空间，单个碎片不能满足作业，但总大小满足

解决方案：紧凑(将碎片合到一起)，系统开销大

动态重定位分区：在可变分区上采用紧凑技术，每次紧凑都需要重定位

多重分区：将作业分成若干片段，采用重定位寄存器确定片段位置

4.4~4.8

分页：将用户程序按逻辑页分成大小相等的片段，叫做页页号起于0，页内地址相对于0

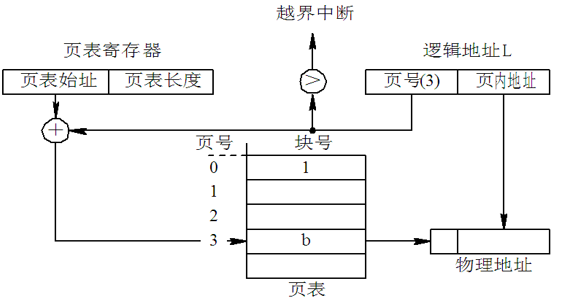
块：内存按页的大小划分成块，大小2的n次幂(512~2k)

分配方式：程序有多少页就分配多少个块，页→页表(由表目→页描述子组成)→块

地址：页号P+位移d，逻辑地址A，页面大小L，P=INT[A/L],d=A MOD L

查找页描述子：=起始地址+页号x页描述子大小，页表存于内存

地址变换：



分配:查询空闲→将总块数，页表始址填入pcb→块号页号填入页表→修改位示图

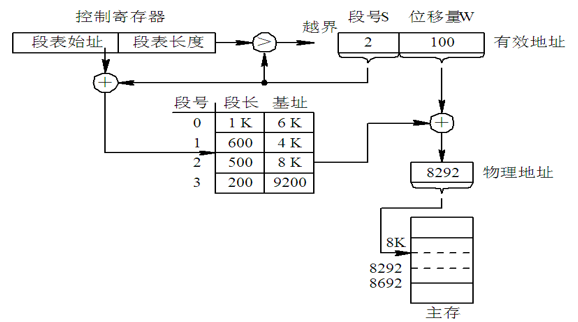
快表：**采用一组硬件寄存器，存放当前访问过的页的页描述子**

**多级页表自己看**

**分段目的：方便编程、信息共享、信息保护、动态增长、动态链接**

**地址：段号+段内地址 段表：段号+段首址+段长**

**硬件：段表始址寄存器+段表长度寄存器**



分页与分段区别：1、页是信息的物理单位，段则是信息的逻辑单位2、页的大小固定且由系统决定，而段的长度却不固定3、分页的作业地址空间是一维的，即单一的线性地址空间，分段的作业地址空间则是二维的

优缺点：便于动态申请、动态链接、共享、管理和使用统一化、产生碎片

**覆盖：将那些不会同时执行的程序段(不一定只有一个作业的)共享同一块内存区域，程序段保存磁盘上，有关程序段的前一部分执行结束，把后续程序段调入内存，覆盖前面的程序段**

**交换：内存空间紧张时，系统将内存中某些进程暂时移到外存，把外存中某些进程换进内存，占据前者所占用的区域，交换→虚拟存储技术**

**交换原则：时间片、优先数等等 何时交换：不用就换出、空间不足换出**

硬件支持：磁盘交换区(足够大、内存能直接存取) 交换位置：受绝对地址产生的时机影响

交换与覆盖的比较：交换不要求用户给出逻辑覆盖结构、交换发生在进程或作业间、覆盖只能覆盖无关程序段

虚拟存储器：具有请求调入功能和自换功能,能从逻辑上对内存容量进行扩充的存储器系统。

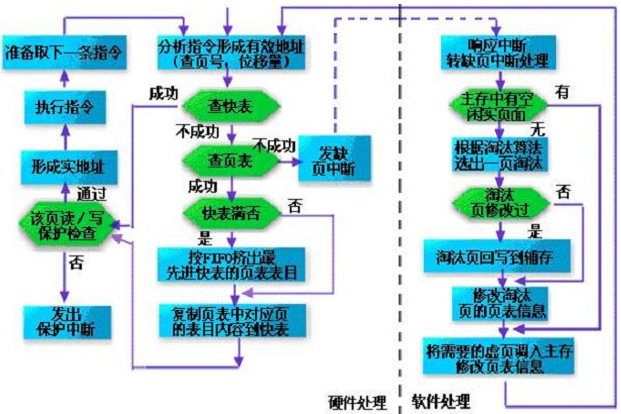
**以CPU时间和外存空间换取昂贵内存空间，离散性、多次性、交换性**

局部性原理：**时间局部性（一条指令被执行了，则在不久的将来它可能再被执行）空间局部性（若某一存储单元被使用，则在一定时间内，与该存储单元相邻的单元可能被使用）**

**请求式分页存储管理：运行→装入0或1个页面→动态装入页面→内存满→淘汰/装入**

**页描述子：页号、中断位(在内/外存)、内存块号、外存地址、访问位(最近被访问过)、修改位(被修改过)、（存取控制、辅存地址） 缺页中断：缺页时请求中断**

**地址变换机制：**



**页面置换算法：**

**最佳置换算法：将未来不会使用或最长时间不被使用的页面替换出，作为参照算法**

**会计算缺页数：内存装满后出现的页框数就是缺页数**

**先进先出：选择在内存中驻留时间最长的页并淘汰，缺页率可能最低**

**最近最久未使用(LRU)：选择最后一次访问时间距离当前时间最长的一页并淘汰之，实现方法→栈或移位寄存器，替换栈底或移位值最大的页面**

**Clock算法：内存所有页面构成循环队列，页面被访问时访问位置1，循环查找页面，访问位为0的页面淘汰，为1置0，按fifo循环查询**

**改进型clock：访问位A(1表示访问过)和修改位M(1表示修改过)，00→最佳置换，01→非最佳，10,11→可能被访问。先查00，失败再查01→（访问位置0），失败将所有访问置0，先查00，再查01**

**缺页中断原理：页不存在→中断调入→有空闲→直接调入/→没空闲→替换**

**缺页次数影响因素：进程的物理页面数，页面大小，程序编译方式，页面淘汰算法**

**颠簸：调度页面所需时间比进程实际运行的时间还多，算法不合理、分配的物理页太少**

**4.9 4.10**

**请求分段数据结构：扩充段表（每个作业一个）、现行分段表(唯一)、现行调用表(唯一)**

**段表：段号 段长 段始址 存取控制 中断位 访问位 修改位 扩充位 辅存地址**

**现行分段表：当前段是否在主存、该段在主存的始地址、共享的作业数、每个作业名等**

**操作：每个允许被共享的段，当其调入主存时，均在此表上登记，当一共享作业退出系统后，则应将共享本段作业数减１并将相应的表目改为“可用”。若共享作业数为０，则请求系统回收相应的存储空间**

**现行调用表：各分段之间有效的、一致的动态连接，调入一分段时，在表中登记段名、访问的入口点、段名及段内地址**

**越界中断处理：需要扩大分段→超出原有的段长→中断→判断扩充位→扩充/出错**

**缺段中断处理：有空闲→调入→修改数据、没有→空闲总和满足→紧凑、不满足→替换**

**为何要进行段的动态链接：多程序段→有的用不着→互斥→执行后不再用→静态开销大**

**动态链接：运行→主程序段进入内存→调用新段→装配、链接主程序**

**页式逻辑地址唯一，难以动态链接**

**间接字：采用间接寻址，间接地址指示的单元的内容，状态位L(1→已建立链接)+直接地址**

**链接中断： L=1→中断、暂停运行→处理→继续运行（L置为0）/L=0→查找直接地址**

**中断处理:间接字→段名/地址→分配段号(缺段与否)→登记段表→修改内存分配表→修改L、直接地址→还原现场→继续运行**

**共享段：因而一个共享段对不同的调用表可以具有不同的段号**

**段页式存储管理：对用户来讲，按段的逻辑关系进行划分；对系统讲，按页划分每一段**

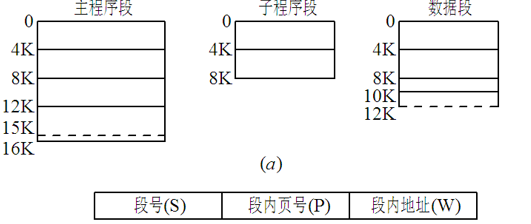


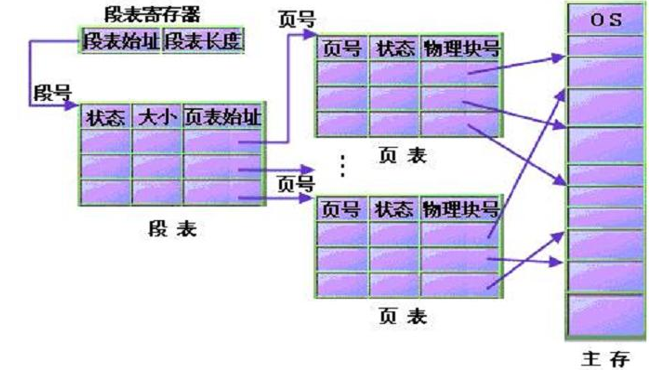
**段表：记录了每一段的页表始址和页表长度**

**页表：记录了逻辑页号与内存块号的对应关系,每段有一个页表，一个程序可能有多个页表**

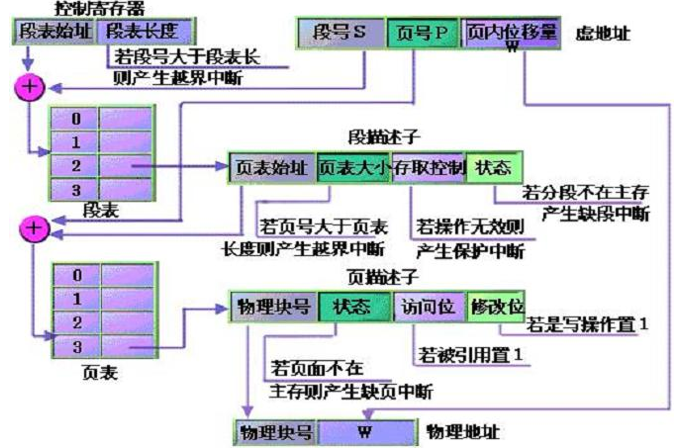
**管理与分配和页式一样**

**地址结构：**

**地址映射**



地址变换



**从控制寄存器读取段表始址，找到段表；**

**段号＋段表始址 得到段描述子地址；**

**从段描述子读取页表始址，找到页表；**

**页号＋页表始址 得到页描述子地址；**

**从页描述子读取物理块号；**

**物理块号＋页内位移量 得到物理地址。**

**地址变换至少要访问主存三次**

**注：段页式看得很烦，需要再理一理**