1. 存储管理
2. 存储体系

按照寄存器->缓存->内存->外存的访问顺序，存取速度越来越慢，容量越来越大，价格越来越低。

1. 地址重定位

可执行文件的建立过程是：源程序->编译->目标模块->链接->可执行文件。当程序执行时由操作系统装入内存而成为进程，程序的逻辑地址被转换成了内存的物理地址，称为地址重定位。

有以下几种方式：

1. 绝对装入

在可执行文件中记录内存地址，装入时直接定位于上述内存地址的方式。程序的地址是在执行之前被确定的，即在编译，链接时指定的内存地址。

优点：过程简单；

缺点：依赖硬件结构，不适合多道程序系统；

1. 可重定位装入

在可执行文件中，列出各个需要重定位点的地址单元和相对地址值，在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，添加相应的偏移量。

（三） 链接

链接是指多个模块在执行时的地址空间分配和相互引用。

1. 静态链接

在生成可执行文件时进行的，在目标模块中记录被调用模块的逻辑地址，在可执行文件中将该地址改写为物理地址。

1. 动态链接

在装入或运行可执行文件时进行的，通常被链接的共享代码称为动态连接库（DDL）。

优点：

多个进程共享DDL；

一个进程可以将多个操作分散在多个DDL中，只将当前的DDL装入内存；

便于代码升级和代码重用；

1. 分区存储管理方案
2. 概述

连续分配存储空间的管理方式，为一个用户程序分配一个连续的内存空间，将内存分为一些大小不等或相等的分区，每个程序可占用多个分区，支持多个程序并发执行，但是难以进行内存分区的共享，因为每个程序需要的内存分区都是不同的，无法调配。因此，有很多碎片产生，内碎片是指在已占用的分区中没利用的空间，外碎片是指占用的分区之间难以利用的空闲分区，一般是小分区。

分区表用于记录分区使用状况，可以只记录空闲分区，也可以同时记录空闲和占用分区。按照分区的划分方式，分为单一连续分区，固定分区和可变分区三种。

（二） 单一连续分区

将整个内存空间的最低端和最高端作为操作系统区，中间作为用户程序区，应用程序可使用用户区的所有空间。

优点： 简单，是用于单用户，单任务系统；

缺点： 一个作业运行要占用整个内存的空间，浪费内存；

（三） 固定分区

为实现多道程序系统，将内存划分为多个固定大小的块的方法，当然，每个区的大小可以不同，但是每个区的大小都是固定的，无法改变。为了便于管理整个内存，需要一个表格来管理，登记了每个分区的大小，起始地址和分配状态。当有作业装入时，系统会好处一个合适的分区。

由于程序要求的内存大小不一定符合分区的大小，因此分区中会有空间浪费，为了防止程序之间的干扰以及非法访问，每个分区都设置了上下界寄存器，限制每个程序访问内存的方位。

优点：与单一连续分区相比提高了内存利用率，可支持多道程序，实现简单，开销小；

缺点：必须预先估计作业大小，内碎片浪费，分区总数固定，限制并发程序数目；

（四） 可变分区

1. 概述

预先不将内存划分为许多分区，而是当作业需要时再向系统申请，从中挖一块分配给该作业。可变分区采用链表的方法管理，优点是没有内碎片，但是依然有外碎片，设置一个空闲分区表，将空闲分区的长度和起始地址放在表中。

1. 分区分配算法

动态分配分区的算法有四种：

1. 最先适应算法

将所有空闲分区按照地址按照地址递增的顺序排列，然后按照分区的先后顺序从头开始查找符合要求的一个分区，如果该分区大于要求的大小，则会产生另一个小分区。在释放分区时，将某个分区归还给操作系统，使其成为空闲区进而被其他作业使用。如果释放的分区与临近的空闲区相连，需要合并为较大的空闲区、

优点：分配简单，尽量保留大的空闲区，满足大内存需求的程序；同时，释放分区可以形成大的空闲区。

缺点：碎片会产生在存储器的各个地方，无法集中使用，降低利用率；

（2） 循环最先适应算法

按照分区的先后次序，从上次分配的分区开始查找，到最后分区时再回到开头，符合要求的第一个分区就是找到的分区。该算法能够使空闲分区分布地更加均匀，但是较大的空闲分区无法保留。

（3）最佳适应算法

将空闲分区按照容量递增排序，在所有大于程序要求的空闲区中挑选一个最小的分区，即最合适的分区。

优点：分配后所剩余的空闲块会最小，较大的空闲区会被保留；

缺点：按照大小排序，因此释放时要在整个链表上搜索地址相邻的空闲区；产生的外碎片非常小；

（4） 最坏适应算法

将空闲分区按照容量递减排序，取最大的一块，将剩余的块再变为更小的空闲区。

优点：只需要查找一次即可；

缺点：剩余的分区越来越小，无法运行大程序；

1. 页式存储
2. 概述

分区存储管理中，如果一个作业的程序空间大于任何一个空闲分区时，就无法装入内存运行。如果我们把作业划分为较小的单位，这些单位可以分散地驻留在内存的碎片中，这样作业就可以装入运行了，这就是页式存储管理，要求进程在内存中不一定连续分配，但是要一次性装入。

1. 原理
2. 虚页和实页

将作业的虚拟地址空间划分为多个长度相等的虚页，每一个程序的虚页都从0开始编号，主存页划分为与虚页长度相同的实页，实页不需要连续，但是一个作业的所有虚页都需要装入内存中，这就要求内存中有足够的空闲块。

1. 页表

分页系统中，页的大小都是2的整数次幂，虚拟地址的结构由两部分组成，P（虚拟页号）和D（页内偏移量）。程序在虚拟地址空间中是连续的，但是在物理地址空间中是不连续的，系统需要在内存中开辟一个页表来建立每个作业的虚页号到物理内存的实页之间的映射，页表中的内容为三部分：页号，块号，页内偏移量。

1. 内存页表

内存页表跟踪记录内存中的实页，物理内存有多少页，内存页表就有多少行，同时还会记录空闲的内存页面数。

1. 进程页表

一个程序的所有页面都是随机分布的，因此需要进程页表来管理，程序装入执行后，需要内存管理单元的支持，根据进程页表进行执行时的动态地址映射和保护，因此进程页表中包含着虚页到实页的映射。

1. 虚页到实页过程说明

虚拟地址分为虚页号和页内偏移量两部分，虚页号和页表控制寄存器中的页表起始之和在累加器中相加得到所要查找的页表中的虚页表项，对应的虚页表项地址为：页表起始地址+虚页号\*页表表项长度；

从页表表项中取出实页号和虚拟地址中的页内偏移量一起装入到地址寄存器中形成内存的物理访问地址；

1. 进程执行的页面变换过程

先得到进程需要的页数，参照内存页表，查看是否有这么多的空闲也，如果有则将N个页面分配给这个进程，并修改内存页表，将程序装入用户区内存，同时建立起这个进程的进程页表，即页表。当一个进程执行完毕退出内存时要撤销进程页表，同时遍历进程页表的每一行，将每一行的物理页号对应到内存页表中进行修改为空闲。

1. 硬件支持
2. 概述

可以看出，每访问一次内存数据，都至少需要访问两次内存，第一次是查找页表，第二次才是真正访问指令或数据。为了加快访问速度，也需要硬件的支持，如页表基址寄存器，页表长度寄存器和联想存储器。

1. 页表基址寄存器

PTBR，每次访问内存时，都会从页表基址寄存器中取出当前进程的进程页表的内存起始地址，然后加上虚页号，就可以访问到当前进程的进程页表中该虚页号对应的实页号，再将实页号加上虚拟地址中的页内偏移量，才能够得到最终的物理地址。在进程切换时，页表基址寄存器的内容才被保存和恢复。

1. 页表长度寄存器

保存了当前进程页表的长度，在每次访问内存之前，需要检查所取得指令的地址是否超过了进程页表的长度，如果超过了则拒绝访问。

1. 联想存储器（快表）

由于每条访问内存的指令，直到需要访问两次内存，因此为了缩短访问时间，可以将页表装入到联想存储器（TLB）中，按照内存来查找虚页到实页的映射。TLB中存储着经常访问的页表表项，以加速地址变换过程，这种页表为快表，内存中的页表为慢表。

联想存储器是一种按照内容进行并行查找的快速存储器，相当于Map，介于内存和寄存器之间的一种存储机制，比内存快很多。由于联想存储器比内存昂贵很多，无法存储所有页表，因此只能装入经常访问的页表表项，在进行地址变换时，变换机构根据虚拟地址中的虚页号同时查找快表和慢表，一旦在快表中找到，则继续执行；否则，用慢表中查到的实页号构造物理地址，同时更新快表，将这一项加入快表中，这也就需要淘汰快表中的某一项了。

1. 分页式存储优缺点

优点：

没有外碎片，内碎片小于页的大小；

程序不必连续存放，便于改变程序占用空间大小，对着程序运行会动态产生数据；

缺点：

程序需要全部装入内存；

动态地址变化会增加计算成本；

各种表格占用空间，以及管理成本；

存储扩充没有解决，如果没有足够的空闲页面来装入整个作业，则作业无法运行；

不易实现共享；

不易实现动态链接；