第二题 可采用哪几种方式将程序装入内存？它们分别适用于何种场合？

一共有三种方式将程序装入内存，分别是绝对装入方式（Absolute Loading Mode），可重定位装入方式（Relocation Loading Mode）和动态运行时装入方式（Dynamic Run-time Loading）。

在计算机系统很小，且仅能运行单道程序时，完全有可能知道程序将驻留在内存的位置，此时可采用绝对装入方式。

当在多道程序的环境下时，编译器不能预知编译后目标模块的的内存位置，需要采用可重定位（又称静态重定位，进程装入后地址不改变）或者动态运行时装入方式。动态运行时装入方式支持进程地址在运行时不断地改变。

第三题 何为静态链接？静态链接时需要解决两个什么问题？

静态链接是指在程序运行之前，先将各目标模块及它们所需的库函数链接成一个完整的装配模块，运行时不再拆开。需要解决两个问题：

1. 对相对地址进行修改，即将编译生成的所有目标模块中的相对地址改为进程统一的相对地址；
2. 变换调用外部符号，即将编译生成的每个模块中所用的外部调用符号也都变换为相对地址。

第六题 在动态分区分配方式中，应如何将各空闲分区链接成空闲分区链？

当进程运行完毕释放内存时，系统根据回收区的首地址，从空闲分区链（表）中找到相应的插入点，判断插入点前后是否有空闲分区：

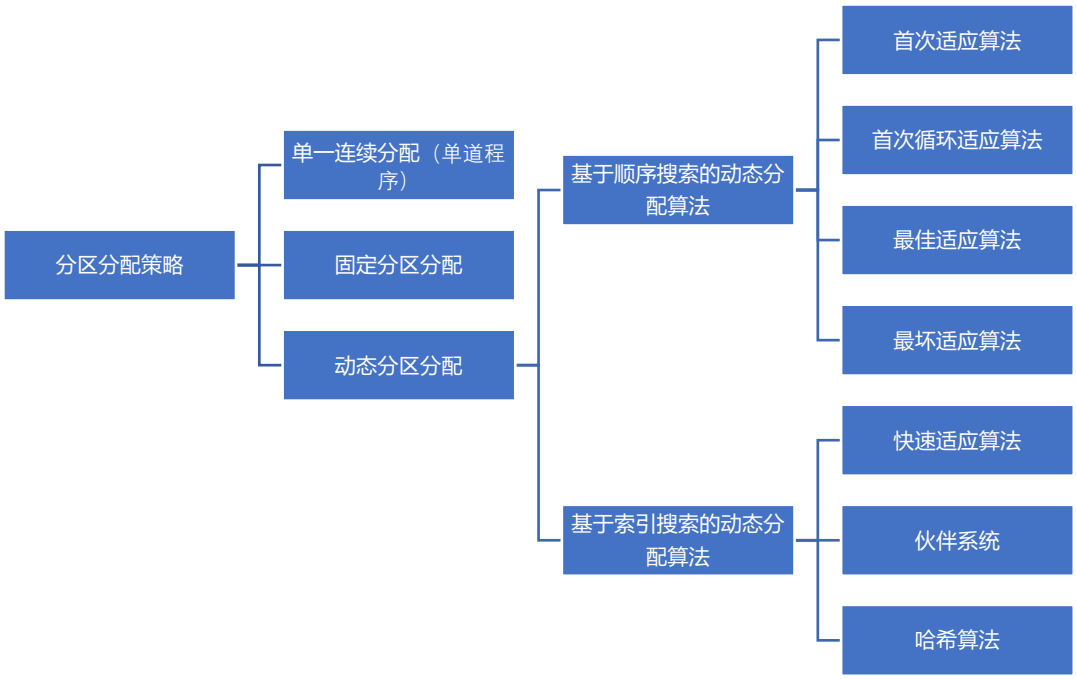
1. 回收区与插入点的前一个空闲分区 F1 相邻接，此时将其与 F1 合并，不必分配新表项，而只需修改 F1 的大小；
2. 回收区与插入点的后一个空闲分区 F2 相邻接，此时将其与 F2 合并，形成新的空闲分区，使用回收区的首地址作为新空闲区的首地址，大小为两者之和，并取消 F2 的表项；
3. 回收区与插入点的前、后两个空闲分区邻接，此时将三个分区合并，形成新的空闲分区，使用 F1 的表项，取消 F2 的表项，分区大小为三者之和；
4. 回收区插入点前后没有空闲分区邻接，此时应为回收区建立一个新表项，填写该分区的首地址和大小，并根据其首地址插入到空闲分区链的适当位置。

第七题 为什么要引入动态重定位？如何实现？

连续分配的方式中，因为一个系统或者一个用户程序必须被放入一段连续的内存空间中。为了利用分割出的小分区，可以使用“紧凑”将小分区合并成大分区。但是每次使用“紧凑”时，都必须对移动之后的程序和数据进行重定位。为了提升效率，可以使用动态重定位分区分配方式规避对所有程序程序和数据进行重定位。

通过在系统中增设一个重定位寄存器，用它来存放程序（数据）在内存中的起始地址。程序执行时，真正访问的内存地址是相对地址与重定位寄存器中的地址相加而成的。当系统对内存进行“紧凑”后，只需将该程序在内存新的起始地址替换掉原来在重定位寄存器中的地址。

第十二题 分区存储管理中常用哪些分配策略？比较它们的优缺点。



单一连续分配：仅适用于单道用户程序运行的系统。

固定分区分配：支持多道用户程序。由于分区大小相同，适用于控制多个相同对象的控制系统；不适用于通用操作系统，造成的内存空间浪费太大。

动态分区分配：支持通用操作系统，有效地提高了系统的内存利用率。

a. 首次适应（first fit, FF）算法

优点：该算法倾向于优先利用内存低址中的小的空闲分区，为以后到达的大作业分配大的空闲分区提供了条件。

缺点：低址部分不断被划分，会留下许多难以利用的、很小的空闲分区，称为碎片。而每次查找空闲分区又是从低址开始查找，又增加了查找空闲分区的时间。

b. 循环首次适应（next fit, NF）算法

优点：可以使内存中的空闲分区变得更均匀，从而减少查找空闲分区时的开销。

缺点：缺乏大的空闲分区。

c. 最佳适应（best fit, BF）算法

优点：该算法第一次找到的空闲分区必然是最佳的。

缺点：因为每次分配后所切割下来的剩余部分总是最小的，会在存储器中留下许多难以利用的碎片。

d. 最坏适应（worst fit, WF）算法

优点：剩下的空闲区不至于太小，产生碎片的可能性最小，对中小作业有利，且查找效率很高。

缺点：存储器中缺乏大的空闲分区。

- - - - - - - - - - - - 以上基于顺序搜索，以下基于索引搜索- - - - - - - - - - - -

e. 快速适应（quick fit）算法

优点：因为不分割空闲分区，所以不产生内存碎片（外零头），能保留大的空间，且查找效率比较高。

缺点：为了有效合并分区，在分区归还主存时的算法复杂，系统开销较大。在被分配的分区存在浪费（内零头）。是一种典型的以空间换时间的做法。

f. 伙伴系统

优点：空间上减少了小的空闲分区，提高了空闲分区的可使用率，优于快速适应算法。时间上比顺序搜索算法好。

缺点：时间上比快速算法差，空间上比顺序搜索算法略差。

g. 哈希算法

优点：查找快速。

缺点：若对空闲分区分类较细，则相应索引表的表项就较多，会显著地增加搜索索引表的表项时间开销。

第二十题 为实现分页存储管理，需要哪些硬件的支持？

对于基本的分页存储管理方式，需要一个页表寄存器（Page-Table Register, PTR），在其中存放页表在内存的始址和页表长度。借助于 PCB，单处理机只需要一个页表寄存器。

对于具有快表的分页存储管理方式，还需增设一个具有并行查询能力的特殊告诉缓冲寄存器，称为联想寄存器（Associative Memory），用于存放当前访问的页表项。

第二十二题 具有快表时是如何实现地址变换的？

在 CPU 给出有效地址后，由地址变换机构自动地将页号送入联想寄存器，并将此页号与高速缓存中的所有页号进行比较，若其中有与此相匹配的页号，便表示所要访问的页表项在快表中。于是可直接从快表中读出该页所对应的物理块号，并送到物理地址寄存器中。如在快表中未找到对应的页表项，则还须再访问内存中的页表，找到后，把从页表项中读出的物理块号送往地址寄存器；同时，再将此页表项存入快表的一个寄存单元中，修改快表。若联想寄存器已满，则用此页表项替换掉一个系统认为不再需要的页表项。

第二十六题 分页和分段存储管理有何区别？

1. 页是信息的物理单位 VS 段是信息的逻辑单位

   分页是系统的行为，对用户来说是不可见的，仅仅是系统提高内存利用率的需要；每段中的信息是相对完整的，主要在于能更好地满足用户的需要。

2. 页的大小固定且由系统决定 VS 段的大小决定于用户编写的程序

   分页是直接由硬件实现的，因而在每个系统中只能有一种大小的页面。而段的长度不固定，通常由编译器在对源程序编译时，根据信息的性质来划分。

3. 分页的用户程序的地址是单一的 VS 分段的用户程序地址是二维的

   分页完全是系统的行为，故分页系统中，用户程序的地址是属于单一的线性地址空间。在分段系统中，用户程序的地址空间是二维的，程序员在标明地址时，要给出段名和段内地址。