1. 静态链接
2. 空间与地址分配
3. 链接器如何将多个输入目标文件的各个段合并到输出文件？

①按序叠加：将输入目标文件按照次序叠加起来。缺点：浪费空间，每个段都有地址和空间对齐要求，x86硬件中，段的装载地址和空间对齐单位是页（4096字节），这样会导致大量内部碎片。②相似段合并：将相同性质的段合并到一起。

1. 两步链接法（Two-pass Linking）？

采用相似段合并空间分配策略的连接器一般采用两步链接法。①空间与地址分配：扫描所有目标文件，获得段长、属性和位置；将个目标文件符号表中的符号定义和符号引用统一存放到全局符号表中；相似段合并，计算各个段合并后的长度和位置，建立映射。②符号解析与重定位：确定“UND”类型符号引用对应符号的目标地址，对重定位入口的地址进行修正

1. 符号地址的确定？

相似段合并后，各个段链接后的虚拟地址（起始地址）就已经确定，因为符号在段内的相对位置固定，链接器为符号加上偏移量就是符号虚拟地址。

1. 符号解析与重定位
2. 重定位表（Relocation Table）、重定位入口（Relocation Entry）？

①每个需要被重定位的ELF段都有一个对应的重定位表，重定位表通常是一个段，因此也被称为重定位段（如.text和.rel.text）。②每个要被重定位的地方叫重定位入口（Relocation Entry）。③重定位入口的偏移（Offset）表示该入口在被重定位段中的位置。④重定位表是一个Elf32\_Rel结构的数组，每个数组元素对应一个重定位入口。

1. 链接时符号未定义（undefined reference to ‘symbol’）？

①链接时缺少某个库；②输入目标文件路径不正确；③符号的声明与定义不一致。

1. 重定位的过程？

每个重定位入口都是一个符号引用，连接器查找全局符号表对符号引用进行重定位，重定位就是确定该符号的目标地址。“UND”类型符号应该能在全局符号表中找到，否则会报符号未定义错误。

1. 指令修正方式的区别？

绝对地址修正后，被修正位置的值为符号的实际虚拟地址；相对地址修正后，被修正位置的值为符号距离被修正位置的地址差。

1. COMMON块
2. 定义的多个符号定义类型不一致如何处理？

①两个强符号不一致，非法；②一个强符号和多个弱符号类型不一致，输出文件中符号所占空间与强符号相同；③两个或两个以上弱符号类型不一致，以占用空间最大的弱符号为准。

1. 为什么不直接把未初始化的全局变量也当作未初始化的局部静态变量一样处理，为它在BSS段分配空间，而是将其标记为COMMON类型的变量？

编译过程中，如果该编译单元中包含弱符号，其所占空间大小是未知的，可能其他编译单元中弱符号所占大小比本单元大，编译器无法为其在BSS段分配空间。链接过程中，所有输入目标文件中的弱符号大小都可以确定，链接器可以在输出文件的BSS段为其分配空间。所以，未初始化的全局变量最终存放在BSS段。

1. C++相关问题
2. 静态库链接
3. 链接过程控制
4. BFD库（Binary File Descriptor library）

一个GNU项目，目的是通过统一接口处理不同目标文件格式，把目标文件抽象成统一的模型，通过操作抽象模型就可以操作所有BFD支持的目标文件格式。

1. 可执行文件的装载与进程
2. 进程虚拟地址空间

①每个进程有独立的虚拟地址空间（Virtual Address Space），空间大小只与CPU位数有关；

②C语言指针大小与虚拟空间位数相同，32位u4字节，64位8字节；

③虚拟地址空间分为内核区和用户区；

④虚拟地址空间不能扩展，物理内存空间可以扩展；通过扩展地址线获得物理内存，通过窗口映射访问内存；Intel称地址扩展方式为PAE（Physical Address Extression）；Windows称访问内存方式为AWE（Adress Windowing Extression），Linux称为mmap()系统调用；

1. 装载的方式

①静态装入；覆盖装入；页映射；

②装载是由操作系统的存储管理器实现的；

1. 从操作系统的角度看可执行文件的加载（创建、装载和执行进程过程）

①创建独立虚拟地址空间：创建将虚存映射到内存的映射函数所需要的数据结构，Linux中称为页目录表(Page Directory)；可以暂时不设置映射关系，如动态运行时加载的请求调页策略，缺页时映射；

②读取可执行文件头，建立虚拟地址空间与可执行文件映射：页表项中的页号和外存地址；

③将CPU寄存器设置成可执行文件入口地址，执行：第①步中入口页面需要动态运行时加载的预调页策略，在加载时就分配内存块设置好映射关系，但是可调出，入口地址在这；

④Linux中将进程虚拟空间中的一个段叫做虚拟内存区域（VMA，Virtual Memory Area），在Windows较虚拟段（Virtual Section）;

⑤除执行预调页策略的页面外，执行请求调页策略的页面只是通过可执行文件的头部信息建立可执行文件到虚存间的映射关系，系统未分配内存块；访问这种页面会处罚页错误（Page Fault），此时系统分配内存，建立映射，重新执行该语句；

1. 进程虚存空间分布

①ELF可执行文件中引入一个概念，Segment包含一个或多个属性类似的Section；一个Segment对应一个VMA；描述Section的结构是段表，描述Segment的结构是程序头（Program Header），记录了ELF可执行文件到虚拟空间的映射；程序头表（Program Header Table）存在于ELF可知性文件和共享库文件，ELF目标文件不装载没有该结构；

②Segment和Section从不同角度划分同一个ELF文件；Section划分ELF文件是链接视图（Linking View）；Segment划分的ELF文件是执行视图（Execution View）；谈论装载时，段指Segment，其他情况，段指Section；

③匿名虚拟内存区域（Anonymous Virtual Memory Area）：在外存中没有映像文件的VMA。堆、栈、虚拟动态共享库和虚拟系统调用都属于AVMA；后两者位于内核区，进程可以访问这些VMA与内核通信；代码VMA（读、执行）、数据VMA（读写、执行）、堆VMA（读写执行）、栈VMA（读写）

1. Linux内核装载ELF过程简介
2. Windows PE的装载