7. 链接

汇编语言是**二进制指令的文本形式**，与指令是一一对应的关系
Linux 中的 as 程序能够将汇编程序翻译成可重定位目标程序
链接可以在程序的生命周期的不同时间发生（链接时、加载时、运行时）
x86-64 代码段总是从地址 **0x400000** 开始，用户栈最大合法用户地址从(2^{48}-1)开始
IA-32 代码段总是从 **0x8048000** 开始，一个页大小为 4kB—4MB 一般为 4KB 可变动。

7.1 静态链接部分

链接器经过**符号解析**和**重定位**两个阶段，将可重定位目标文件生成可执行目标文件

在**符号解析**步骤中，链接器将每个**符号引用**与一个确定的**符号定义**关联起来
重定位分三步：1.合并相同的节 2.对定义符号进行重定位 3.对引用符号进行重定位

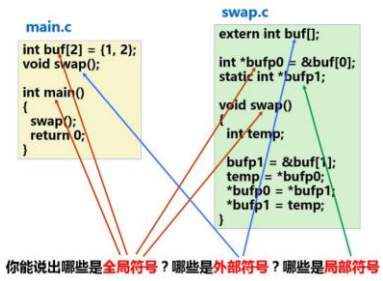
三种目标文件：

1. **可重定位目标文件**（.o）：每个.o 文件的代码和数据地址都从 0 开始
   可重定位目标文件可以合成静态库文件(.a)
2. **可执行目标文件**（a.out）：代码和数据地址为**虚拟地址空间**中的地址
3. **共享目标文件**（.so）

Linux 目标文件标准二进制格式：**ELF 格式**（可执行可链接格式）
ELF 格式下的两种视图：

1. **链接视图**（需要被链接，各种节组合成）：就是可重定位目标文件（.o）
2. **执行视图**（需要被执行，各种段组合成）：就是可执行目标文件（a.out）

链接符号的类型：
静态变量 static 都在**全局数据区**分配内存，包括静态局部变量,都属于**局部符号**

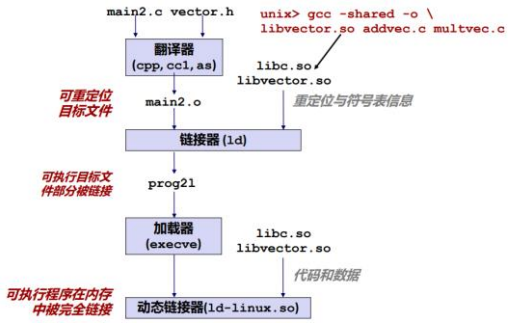


1. **全局符号**：由模块 m 定义的，可以被其他模块引用的符号
   例如：**非静态** C 函数与**非静态**全局变量（main.c 中的全局变量名 buf）
2. **外部符号**：由模块 m 引用的全局符号，但由其他模块定义
3. **本地/局部符号**：由模块 m 定义和仅由 m 唯一引用的符号
   如：使用**静态属性定义**的 C 函数和全局/局部变量
   本地链接符号不包括非静态本地局部变量(存储在栈上)
   本地的静态 C 变量（static）：存储在 .bss 或 .data

全局符号的强弱特性：强符号：**函数名和初始化全局变量**　　弱符号：**未初始化的全局变量**
符号处理规则：

1. 不允许多个同名的强符号（每个强符号只能定义一次，否则：**链接器报错**）
2. 若有一个强符号和多个弱符号同名，则选择强符号（数据和大小按强符号定义为准）
3. 如果有多个弱符号定义，以**任一个**为基准（输出警告信息）

7.2 动态链接部分



没有任何共享库文件复制到可执行文件中，但在链接时复制了共享库文件的一些符号表和重定位信息用来解析引用

动态链接可以**加载时链接**和**运行时链接（dlopen 函数）**。
加载时链接由**动态链接器(ld-linux.so)**自动处理，标准 C 库（**libc.so**)通常采用**动态链接**。
共享库是一个目标文件，在加载和运行时可以加载到**任意内存地址**，共享库中的.text 节的副本可以被不同运行的进程共享。
共享库中的代码为位置无关代码 PIC（可以加载而无需重定位的外码）
共享库的所有引用情况：

1. 模块内部的过程调用、跳转（采用 PC 相对偏移寻址，在动态库中相对位置不变）
2. 模块内部数据访问（对动态库模块中全局变量和静态变量的访问）
3. 外模块对动态库中的过程调用，跳转（需要 **PIC**）
4. 外模块对动态库中的数据访问（需要 **PIC**）

在程序头部表的引导下，**加载器**将可执行文件的片复制到代码段和数据段（缺页至少两次，还有栈、堆所以至少 4 次）

程序启动并加载过程：程序启动./hello → 调用 fork() → 以构造的 argv 和 envp 为参数调用 execve() → execve()调用加载器进行可执行文件加载，最终转去执行 main

子进程通过 execve 调用启动**加载器**，加载器删除子进程现有的内存虚拟段，并创建一组新的只读代码段、读写数据段、堆段和栈段，新的堆和栈段被初始化为 0，在加载过程中没有任何从磁盘到内存的数据复制，直到 CPU 引用一个被映射的虚拟页时才会进行复制（缺页处理，缺页至少 4 次）。

库打桩机制：允许程序员截获对任意函数的调用，取而代之执行自己的代码
打桩可出现在（**编译时**、链接时、加载/运行时【需动态链接】）

C 语言函数中的整数常量都存放在程序虚拟地址空间的**代码/数据段**
C 语句中的全局变量，在**链接**阶段被定位到一个确定的内存地址
链接过程中，带 static 属性的全局变量属于**局部符号**
简述 C 编译过程对非寄存器实现的 int 全局变量与非静态 int 局部变量处理的区别。包括存储区域、赋初值、生命周期、指令中寻址方式等

什么是共享库（动态连接库）？简述动态链接的实现方法
答：共享库（动态连接库）是一个.so 的目标模块（elf 文件），在运行或加载时，由动态链接器程序加载到任意的内存地址，并和一个和内存中的程序（如当前可执行目标文件）动态完全链接为一个可执行程序。使用它可节省内存与硬盘空间，方便软件的更新升级。如标准 C 库 libc.so 。
加载时动态链接：应用程序第一次加载和运行时，通过 ld-linux.so 动态链接器重定位动态库的代码和数据到某个内存段，再重定位当前应用程序中对共享库定义的符号的引用，然后将控制传递给应用程序（此后共享库位置固定了并不变）。
运行时动态链接：在程序执行过程中，通过 dlopen/dlsym 函数加载和连接共享库，实现符号重定位，通过 dlclose 卸载动态库。