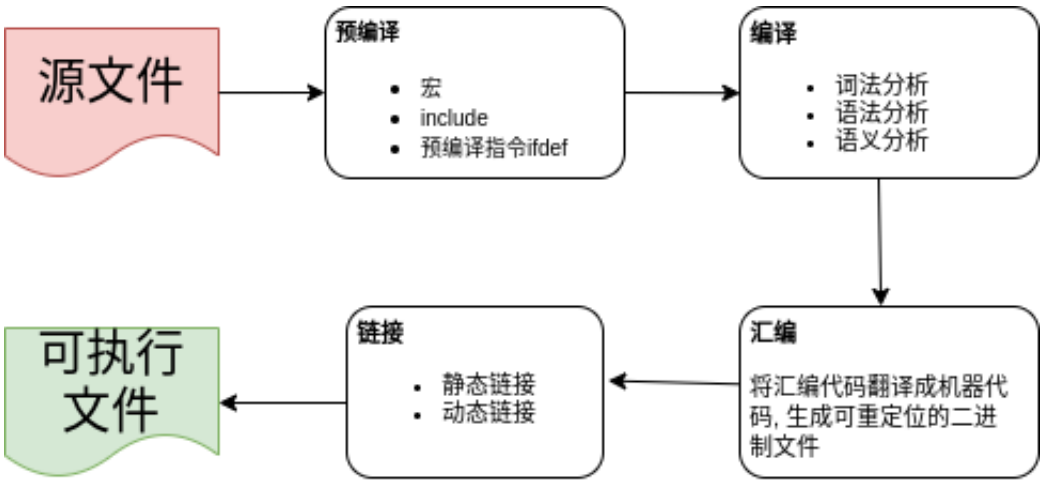
静态链接和动态链接

MengChunlei

January 10, 2021

1 编译过程

从源文件到可执行文件, 分为四个步骤, 如下图所示:



2 链接的原因

程序开发的过程中, 不会把所有的代码都跟主函数写在同一个 *.cc* 文件中, 而是不同的模块功能实现在不同的文件中, 模块间的依赖以及主函数文件对其他模块的依赖通过 *include* 对应模块的头文件来实现. 而在预编译 → 编译 → 汇编后, 对于每个模块, 生成了自己对应的目标文件. 最后需要将这些目标文件跟主函数所对应的目标文件" 整合" 到一起生成可执行文件, 这个过程就是链接.

3 静态链接

3.1 原理

静态链接就是在生成可执行文件的时候, 所有的目标文件经过重定位等一系列操作包含在最后的可执行文件中. 假设有下面的源文件:

Listing 1: hello.h

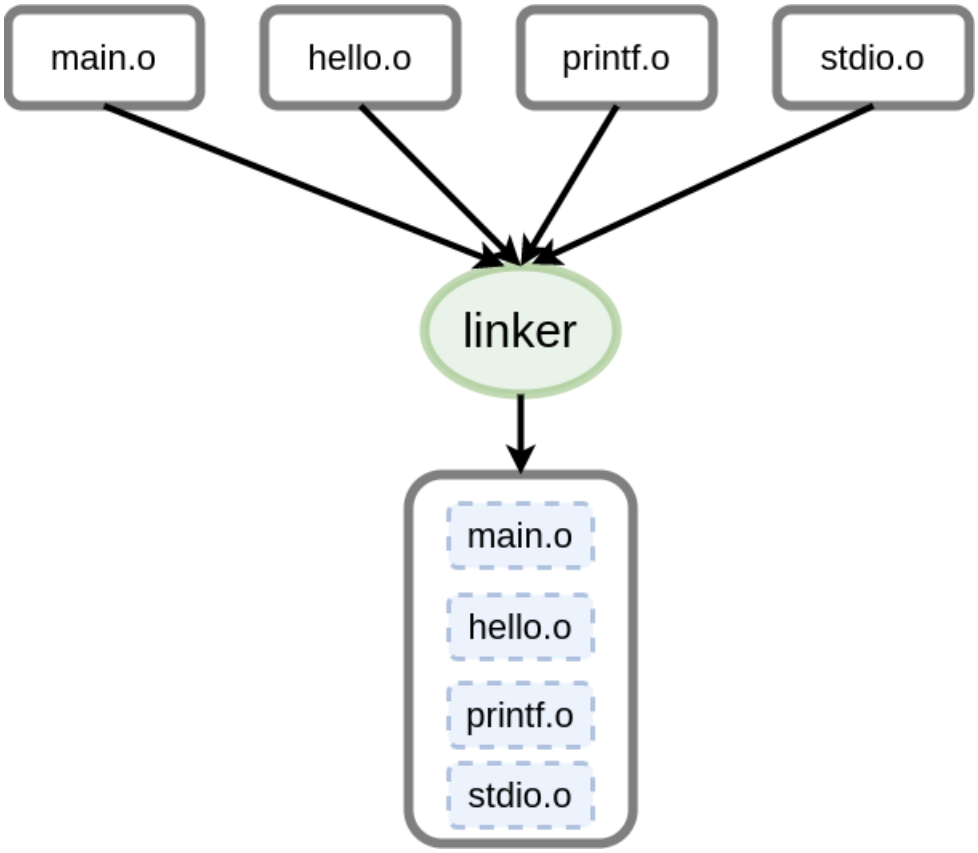
```
inline int Add(int a, int b) {
  return a + b;
}
```

Listing 2: main.cc

```
#include <stdio.h>
#include "hello.h"

int main() {
  printf("Hello World\n");
  return 0;
}
```

那么生成的可执行文件如下图所示. 最后的可执行文件包含了所有使用到的目标文件.



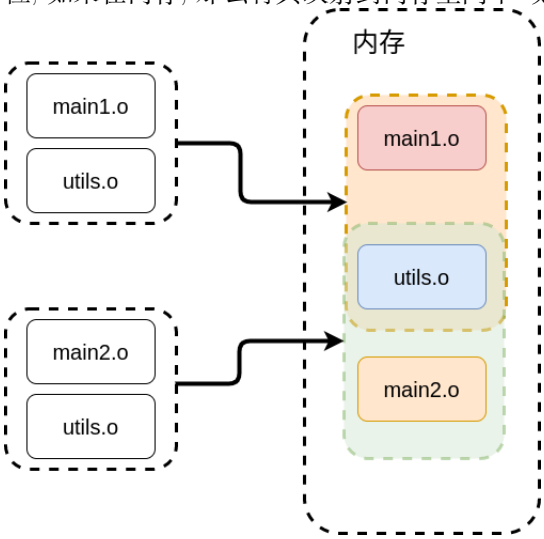
3.2 优缺点

- 优点是可移植性好, 因为不再依赖其他文件
- 第一个缺点文件太大
- 第二个缺点更新麻烦, 如果一个模块更新, 那么要重新链接生成可执行文件
- 第三个缺点如过一个模块被静态链接到多个可执行文件中, 那么在运行这些程序的时候, 这一个模块不能共享从而会占用很多内存

4 动态链接

4.1 原理

动态链接是在运行的时候才把它使用的动态链接库进重定位等操作. 而可执行文件在磁盘上的时候是不包含实际的目标文件的. 在加载的时候去查找使用的动态链接库, 如果不在内存中, 那么将其加载到内存, 并且进行重定位; 如果在内存, 那么将其映射到内存空间中. 如下图所示:



4.2 优缺点

- 第一个优点文件磁盘上的文件变小
- 第二个优点易更新
- 第三个优点可共享内存
- 第一个缺点移植性不好, 从一台拷贝到另一台机器的时候, 需要所有的动态链接库的位置一样
- 第二个缺点性能有所损失, 因为要在运行的时候进行一些重定位操作