LinkLab 报告

姓名：赖柏有
学号：2022201388

Part A: 思路简述

本实验旨在设计和实现一个简单的链接器，主要功能包括符号解析、重定位和段合并。链接器的核心思想是逐个处理输入的目标文件，合并其各个段（如 .text、.data 等），并解析全局符号。重定位主要针对地址相关的符号进行计算，确保目标文件的符号地址与最终执行文件的虚拟地址相符。

关键的数据结构包括符号表、段表和重定位表。符号表存储所有全局符号及其地址信息；段表管理程序的各个段及其数据；重定位表存储重定位项，确保符号的地址在合并后的文件中得到正确处理。

Part B: 具体实现分析

符号解析实现

在符号解析的过程中，我们首先根据符号类型将符号分为全局符号、局部符号和弱符号。对于全局符号，若在多个目标文件中出现，则检查是否存在强符号与弱符号冲突，确保使用强符号的定义。局部符号仅在所在目标文件内部使用，通过添加目标文件名后缀来避免与其他目标文件中的符号冲突。弱符号优先使用强符号定义。

对于符号冲突的解决，我们采取了两种策略：若存在强符号与弱符号的冲突，则优先选用强符号；若多个强符号定义相同符号，则抛出错误，提示用户存在重复定义。

在符号解析的过程中，错误检查尤为重要。例如，若在重定位时找不到目标符号，我们会抛出 "Undefined symbol" 错误。

重定位处理

本实验支持的重定位类型包括 R_X86_64_32、R_X86_64_32S 和 R_X86_64_PC32。针对不同的重定位类型，重定位算法采用不同的处理方式：

对于 R_X86_64_32，将符号的地址填入对应位置，截断为 32 位。
对于 R_X86_64_32S，处理类似，但需要确保符号位的扩展是合法的（符号扩展）。
对于 R_X86_64_PC32，计算相对偏移量，考虑程序计数器 %rip 的位置，确保跳转指令正确。
错误处理方面，我们会检查重定位符号是否已定义，若未定义则抛出 Undefined symbol 错误。此外，重定位过程中需要进行合法性检查，确保所有计算结果符合目标平台的要求。

段合并策略

在合并段的过程中，我们首先按段名进行合并。例如，.text 和 .rodata 等段会被合并到目标文件的相应位置。我们特别注意段的对齐，确保每个段的起始地址是 4KB 对齐，以符合常见操作系统的内存管理要求。

局部符号的存储考虑到其只在本目标文件内有效，因此我们采用了符号名称后加目标文件名后缀的方法来避免与全局符号冲突。

Part C: 关键难点解决

关键难点是第三步重定位，起初test_3一直发生段错误但是找了很久都找不到重定位是哪里出错了，后面看了很久才知道是没有考虑到P是要根据%rip计数器进行调整的，修改了这个才终于把这一关过了

Part D: 实验反馈

<!-- 芝士 5202 年研发的船新实验，你的反馈对我们至关重要
可以从实验设计，实验文档，框架代码三个方面进行反馈，具体衡量：

1. 实验设计：实验难度是否合适，实验工作量是否合理，是否让你更加理解链接器，链接器够不够有趣
2. 实验文档：文档是否清晰，哪些地方需要补充说明
3. 框架代码：框架代码是否易于理解，接口设计是否合理，实验中遇到的框架代码的问题（请引用在 repo 中你提出的 issue）
   -->
   实验估计7-
   15个小时但对于我来说，靠AI和自己硬磕搞了可能二十多个小时都没有怎么搞好，最后还是参考了别人的代码才能完成，我觉得有可能是我自己本身编程能力太烂了，但这个学习的过程确实对链接器更懂了，虽然自己上手写的时候真的写不好

参考资料 （可不填）

<!-- 对实现有实质帮助的资料 -->

1. [资料名] - [链接] - [具体帮助点]
2. ...