4.27上机课

实验目标

使用binutils工具来观察ELF文件

学习如何利用Linux系统提供的二进制工具集分析可执行文件的结构和依赖关系

学会写动态链接库并链接指定的动态库

掌握动态链接库的创建过程以及如何将应用程序与自定义动态库链接

会针对动态链接函数来Inerposition

了解如何拦截和替换动态链接库中的函数调用，实现性能监测等高级功能

实验内容

提供的文件

- 可执行文件demo
- 动态链接库libseries.so

内容

1. 使用binutils工具观察ELF文件
2. 编写高效的动态链接库
3. 实现函数Interposition并测量性能

提交

一份包含以下内容的文档：

- 观察报告（包含使用的命令和观察到的信息）
- 动态链接库代码和说明
- Interposition代码和现象解释

详细步骤

使用binutils工具观察ELF文件

1. 使用ldd观察demo可执行文件：需要确定demo链接了哪些动态库以及存在哪些undefined symbol
2. 使用readelf读取ELF格式文件信息：分析demo和libseries.so的动态链接信息、headers和sections
3. 使用objdump反汇编文件：反汇编demo和libseries.so，配合readelf读取动态链接函数的GOT
4. 分析函数原型和功能：通过阅读汇编代码，生成两个函数的原型并逆向分析其功能

编写高效动态链接库

1. 分析现有函数：通过逆向工程理解libseries.so中实现的函数功能和算法
2. 设计高效算法：根据数学知识，设计更加高效的计算方法替代原有实现
3. 编写动态链接库：实现优化后的函数并编译生成新的动态链接库
4. 测试运行demo：让demo链接新编写的动态链接库并验证功能正确性

实现函数Interposition

根据课程学到的知识，在执行demo时记录libseries.so和你自己编写的动态链接库函数被demo调用时执行需要的时间

参考实现：

1. 添加计时功能：使用gettimeofday()或clock_gettime()函数实现微秒或纳秒级的精确计时（需要<sys/time.h>或者<time.h>，POSIX系统支持）
2. 编写Wrapper函数：创建包装原始函数的新函数，在调用前后添加计时代码
3. 设置链接优先级：确保Wrapper函数在链接时优先于原始库函数被选择
4. 记录执行时间：比较并记录原始libseries.so和自己编写的动态链接库函数的执行时间