

**HUAZHONG AGRICULTURAL UNIVERSITY**

hzau-1

综合实训日志

|  |  |  |  |
| --- | --- | --- | --- |
| 题 目 | ELF 文件解析器的设计与实现 | | |
| 姓 名 | 詹少雄 | 学 号 | 2019308210202 |
| 专业班级 |  |  |  |
| 计科2003班 | | |
| 指导老师 | 任继平 |  | 李小霞 |

中国·武汉

2023 年 6 月

# 日期：2023年6月19日

**问题提出与分析：**

今天是实训的第一天，首先我们进行了小组组队，在开始进行分析前，我们首先学习了相关的知识，然后我们进行了小组分工等任务分配。什么是ELF文件？ELF文件的作用是什么？

**组内讨论记录：**

小组分工情况初步确定，初步分工情况如下：

* 詹少雄（计科2003）

负责统筹团队任务，安排实训方案与讲解，并成功完成了-h、-S、-l和-d的ELF解析选项。在项目中展示了出色的组织能力和解析技术，为团队提供了方向和指导。

* 李绅（计科2004）

负责实训方案的实行与讲解，并顺利完成了-n、-e、-r和-d的ELF解析选项。展现了良好的实施能力和解析技术，为团队提供了有关ELF文件的重要信息。

* 杜鹏（计科2003）

负责实训方案的实行与讲解，并圆满完成了-s、-t、-g和-d的ELF解析选项。在项目中展现了优秀的编程能力和解析技术，为团队提供了强大的功能。

**讨论总结**（包括但不仅限于问题解决方案，如分工情况、算法、系统设计方案等）：

本日完成任务：

1. 完成ELF文件格式的学习；
2. 进行《程序员的自我修养》ELF部分的学习
3. 小组分工与任务计划
4. 创建git仓库，便于小组合作

**实验过程记录**（包括但不仅限于实验方案设计、实验结果记录等）：

我们创建了Github仓库，完成团队协作工作流的过程

我们采用Git工作流，仓库开源于：

<https://github.com/Jasaxion/LinuxELF_FileParse>

欢迎访问！

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图 1 我们开源的Github仓库截图

我们使用Git工作流，将工程项目进行开源共享，并且更加方便我们团队成员进行修改和溯源，采用Github工作流对于我们的项目来说具有重要意义。它提供了一个集中、协作和可追踪的平台，使得团队成员可以更加高效地合作，管理和追踪项目的进展，并与其他开发者进行交流和分享。通过这种方式，我们能够更好地组织和管理我们的项目，提高工作效率，以及促进项目的成功完成。

# 日期：2023年6月20日

**问题提出与分析**：

今天是实训的第二天，基于我们昨日进行的分工情况，我负责完成了-h的头文件信息展示的部分，什么是ELF头文件信息？Magic数是什么？文件头的其他成员参数有哪些？

**组内讨论记录**：

通过查阅相关的资料以及阅读《程序员的自我修养》

每种可执行文件的格式的开头几个字节都是很特殊的，特别是开头4个字节，通常被称为魔数（Magic Number）。通过对魔数的判断可以确定文件的格式和类型。如：ELF的可执行文件格式的头4个字节为0x7F、e、l、f；Java的可执行文件格式的头4个字节为c、a、f、e；如果被执行的是Shell脚本或perl、python等解释型语言的脚本，那么它的第一行往往是#!/bin/sh或#!/usr/bin/perl或#!/usr/bin/python，此时前两个字节#和!就构成了魔数，系统一旦判断到这两个字节，就对后面的字符串进行解析，以确定具体的解释程序路径。

**讨论总结**（包括但不仅限于问题解决方案，如分工情况、算法、系统设计方案等）：

通过小组讨论，我详细了解到了Magic数的含义

* 第1个字节：0x7f 表示ELF标记
* 第2个字节：‘E’
* 第3个字节：‘L’
* 第4个字节：‘F’
* 第5个字节：ELF文件类型，也就是class
* 第6个字节：数据编码信息，字节序信息，大端序、小端序
* 第7个字节：ELF版本号：标识ELF Version, 该值等于EV\_CURRENT
* 第8个字节：OS信息
* 第9个字节：ABI 版本信息
* 第10~16个字节：填充位

**实验过程记录**（包括但不仅限于实验方案设计、实验结果记录等）：

图示

描述已自动生成

图 2 -h选项程序流程图

通过今天的学习和讨论，我完成了-h选项的分析和代码编写，如上图所示，其为我实现-h文件头信息的程序流程图。



图 3 文件头信息的数据结构定义「以32位进行举例」

# 日期：2023年6月21日

**问题提出与分析：**

今天是实训的第三天，昨天我完成了-h选项文件头信息的解析和代码编写，在今天我计算完成-S节区表信息的分析解析和代码编写，需要问答几个问题，什么是节区表？节区表有哪些成员参数，这些成员参数分别有何定义和作用？

**组内讨论记录：**

通过查阅相关的资料以及阅读《程序员的自我修养》以及进行小组讨论。

段（segment) 和 节（section)是有区别的，节不是段。段是程序执行的必要组成部分。 在每个段中会有代码或者数据被划分为不同的节。 而节头表则是对这些节的位置和大小的描述，主要是用于链接和调试。

**讨论总结**（包括但不仅限于问题解决方案，如分工情况、算法、系统设计方案等）：

首先，我们通过集体讨论的方式对-S节区表的所有成员进行了详细的分析和阐述。我们深入研究了每个节区的含义、作用以及与程序的链接和执行过程的关系。

我们从节区表的开始位置开始，逐个解析和讨论了每个节区的条目。我们仔细研究了节区的名称、类型、大小、偏移量和属性等信息，并分析了它们在程序中的作用。

包含的成员参数如下：



图 4 成员参数图1

表格

描述已自动生成

图 5成员参数图2

**实验过程记录**（包括但不仅限于实验方案设计、实验结果记录等）：

文本

描述已自动生成

图 6程序头信息的数据结构定义「以32位进行举例」

图示

描述已自动生成

图 7 -S选项程序流程图

-S 选项用于显示ELF文件中的节区信息。其实现步骤如下:1. 解析ELF头部,获取节区头表的位置和大小。ELF头部中包含节区头表的起始地址和大小。2. 读取节区头表。节区头表由多个Elf32\_Shdr或Elf64\_Shdr结构组成,每个结构描述一个节区的信息。3. 解析每个节区头,获取节区名、类型、地址、偏移、大小等信息。4. 打印每个节区的信息。按照节区类型和地址顺序打印。对于每个节区,打印其节区名、类型、地址、偏移、大小等信息。5. 打印节区内容的摘要信息。对于典型的节区类型,如.text、.data、.bss等,打印其中的符号数量和大小信息。6. 打印编译器版本信息。如果文件包含.comment节区,从中解析出GCC版本信息并打印。7. 打印动态链接信息。如果文件是动态链接库,从.dynamic节区解析动态链接信息并打印。以上就是readelf -S 选项实现的主要步骤。它通过解析ELF文件中的各种节区,获取并打印节区信息,为分析和调试ELF文件提供帮助。

# 日期：2023年6月23日

**问题提出与分析**：

今天是实训的第四天，前段时间我我完成了-h和-S选项的深度解析和代码编写，在今天我计划完成-l程序头表信息的分析解析和代码编写，需要问答几个问题，在执行视图的程序头表中段的含义是什么？程序段有什么作用？用得最多的几个段结构是什么？

**组内讨论记录**：

ELF 程序头是对二进制文件中段的描述，是程序装载必须的一部分。

* 段（segment) 是在内核装载时被解析的。
  + 主要作用就是描述磁盘上可执行文件的内存布局以及如何映射到内存中。
  + 可以通过引用原始的ELF头中名为： e\_phoff(程序头表的偏移量)的偏移量来得到程序头表。

段的作用是什么呢？

1. 显示程序头表信息。程序头表描述ELF文件各个段(代码段、数据段等)在文件中的布局信息。
2. 显示节区头表信息。节区头表描述ELF文件各个节区(代码节、数据节等)在文件中的布局信息。
3. 帮助分析ELF文件 segments和sections的组织结构。

**讨论总结**（包括但不仅限于问题解决方案，如分工情况、算法、系统设计方案等）：

对于这一部分的内容，我通过查阅资料与此同时与小组成员进行分析讨论，得出了以下的结论

程序头表显示如下信息：

1. p\_type：表示程序段的类型信息，标识了段的类型
   1. 主要类型包括PT\_LOAD，PT\_DYNAMIC，PT\_INFERP
      1. PT\_LOAD类型的段会在创建进程时加载到内存中
      2. PT\_INFERP类型的段包含了.interp节，该节提供了加载二进制文件的解释器的名称
      3. PT\_DYNAMIC包含了.dynamic节，该节告诉解释器如何解析二进制文件用于执行
2. p\_offset：表示分段文件的偏移信息
3. p\_vaddr：表示段段虚拟地址信息
4. p\_paddr：表示段物理地址信息
5. p\_filesz：文件中段的大小
6. p\_memsz：内存中段段大小
7. p\_flags：段的标志信息
   1. 指示了段在运行时的访问权限，有三种重要的类型：
      1. PF\_X（可执行），PF\_W（可写），PF\_R（可读）
8. p\_align：段、文件、内存对齐信息：指定了段所需的内存对齐方式（字节为单位）
   1. 提供了二进制文件的段视图，ELF包括零或多个节，实际上就是把这些节捆绑成单个块
   2. 段提供的可执行视图，只有二进制文件会用到

今天提出了关于ELF文件解析器的一些问题：

1. 为什么需要进行大小端转化等等

我们将收集到的问题与老师进行了详细的讨论，并设计出了解决方案。

关于我们的讨论情况，我们使用Github工作流将其上传到Github的issue部分，这样我们3个团队成员便可以进行交流讨论，共同进行讨论和设计解决方案。

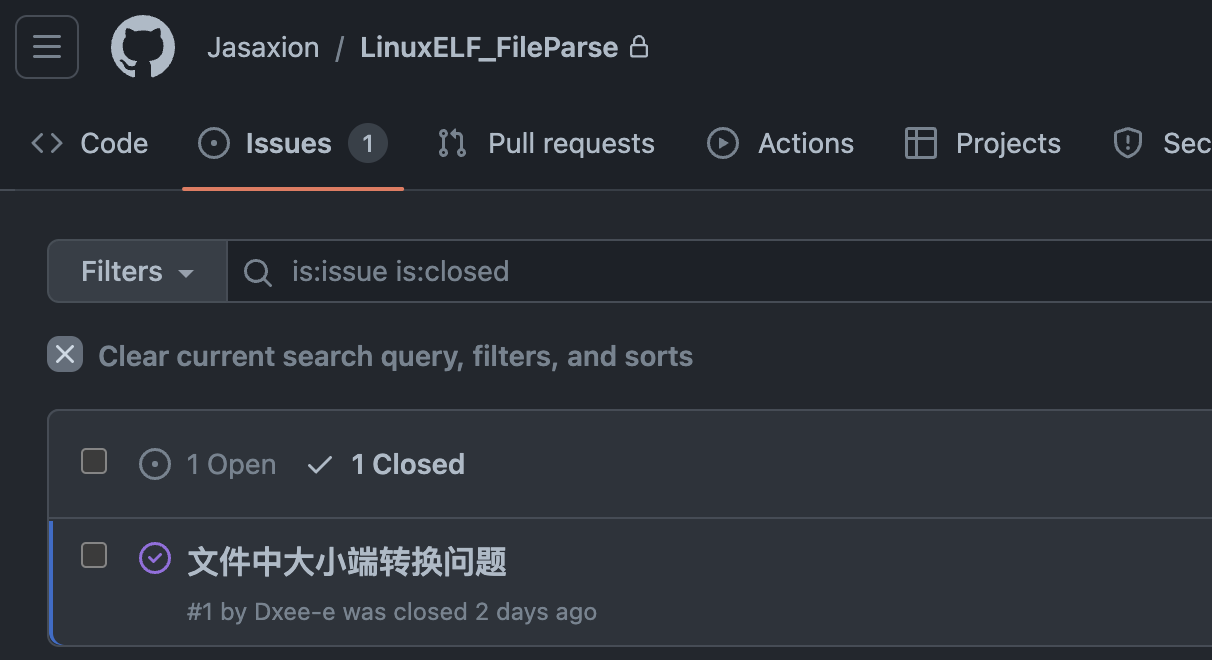
****

图 8 有问必答，共同进行讨论解决问题

**实验过程记录**（包括但不仅限于实验方案设计、实验结果记录等）：

文本

描述已自动生成

图 9 -l选项查看程序头段表信息图「以32位进行举例」

图示

描述已自动生成

图 10-l选项程序实现流程图

实现步骤简述：

1. 解析ELF头部,获取程序头表和节区头表的位置和大小。ELF头部包含两个表的起始地址和大小。
2. 读取程序头表,解析每个程序头,打印程序头信息。信息包括类型、偏移、虚拟地址、物理地址、大小等。
3. 读取节区头表,解析每个节区头,打印节区头信息。信息包括名称、类型、地址、偏移、大小等。
4. 根据需要打印额外信息,如节区内容摘要信息等。
5. 如果文件是动态库,还需要解析和打印.dynamic节区中的动态链接信息。
6. 打印编译器版本信息(如果有.comment节区)。

# 日期：2023年6月24日

**问题提出与分析**：

今天是实训的第五天，前一段时间基本完成了对于-h、-S、-l选项的分析和应用，同时完成了代码的撰写工作，与此同时，团队成员也都分别完成了各自的独立的任务，今天，我们团队三个人共同进行小组讨论和分析，计划对于-d选项进行分析，提出几个问题？如何实现动态链接的数据结构？动态链接和静态链接的差异在哪？动态链接的基本原理是什么？方法是什么？

**组内讨论记录**：

* **静态链接**

事实上，静态链接的过程就是上文所描述的过程。在Linux中，静态链接器（static linker）ld以一组可重定位目标文件和命令行参数作为输入，生成一个完全链接的、可以加载和运行的可执行目标文件作为输出。输入的可重定位目标文件由各种不同的节组成，每一节都是一个连续的字节序列。

* **动态链接**

静态链接使得进行模块化开发，大大提供了程序的开发效率。随着，程序规模的扩大，静态链接的诸多缺点也逐渐暴露出来，如：浪费内存和磁盘空间、模块更新困难等。在静态链接中，C语言静态库是很典型的浪费空间的例子。关于模块更新，静态链接的程序有任何更新，都必须重新编译链接，用户则需要重新下载安装该程序。

解决空间浪费和更新困难最简单的方法便是将程序的模块相互分割开来，形成独立文件。简而言之，就是不对那些组成程序的目标文件进行链接，而是等到程序要运行时才进行链接。

**讨论总结**（包括但不仅限于问题解决方案，如分工情况、算法、系统设计方案等）：

我们小组三个人一同进行讨论设计了动态链接共享库的分析和过程

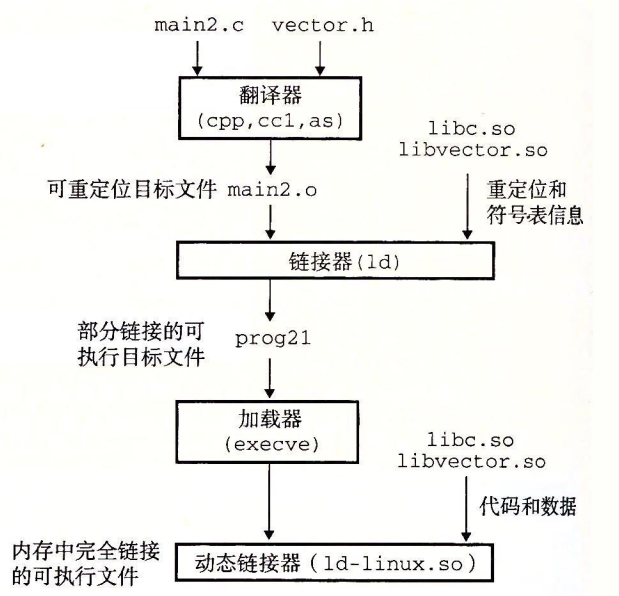


图 11动态链接共享库过程图

动态节区表保存了动态链接器所需要的基本信息，比如依赖于哪些共享对象、动态链接符号表的位置、动态链接重定位表的位置、共享对象初始化代码的地址等.

* d\_tag：表明了动态表项的类型。以下是一些常见的类型:
  + DT\_SYMTAB：表示动态链接符号表的地址，d\_ptr为.dynsym的地址
  + DT\_STRTAB：表示动态链接字符串表地址，d\_ptr为.dynstr的地址
  + DT\_STRSZ：动态链接字符串表大小，d\_val为大小
  + DT\_NEED：依赖的共享对象文件，d\_ptr表示依赖的共享对象文件名
  + DT\_REL：动态链接重定位表位置
* d\_val or d\_ptr：是一个联合体，要么表示一个值，要么表示一个地址，具体的情况因动态表的类型不同而不同

链接过程主要包含了三个步骤：

1. 地址与空间分配（Address and Storage Allocation）
2. 符号解析（Symbol Resolution）
3. 重定位（Relocation）

**实验过程记录**（包括但不仅限于实验方案设计、实验结果记录等）：

****

图 12-d查看动态链接信息图「以32位进行举例」

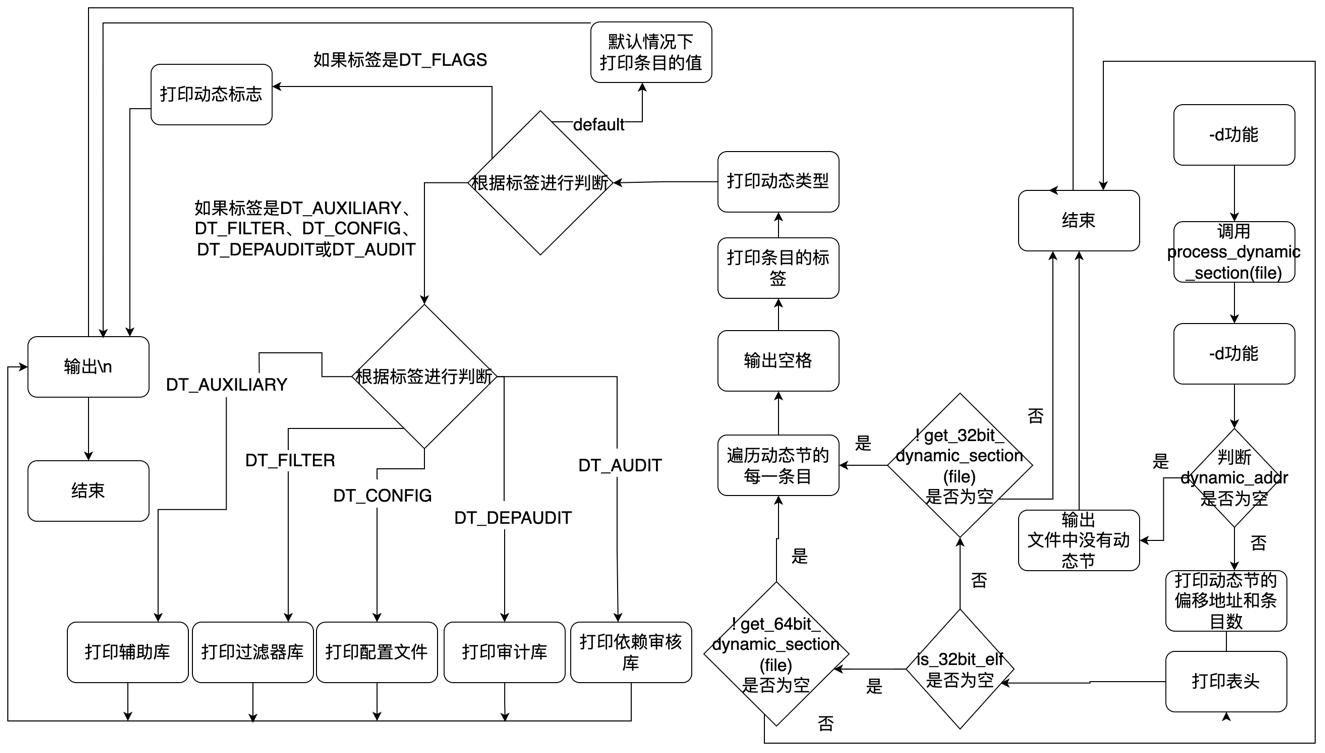


图 13-d选项程序实现流程图

1. 定义一个名为 ELF\_process的类的成员函数 get\_32bit\_dynamic\_section，参数为一个文件指针 file。

2. 分配内存，用于存储32位动态节的外部表示，将其转换为 Elf32\_External\_Dyn类型的指针 edyn。

3. 定义一个指向32位动态节外部表示的指针 ext、一个指向32位动态节的指针 entry。

4. 将文件指针定位到动态节的偏移地址处。

5. 从文件中读取32位动态节数据到 edyn内存块中。

6. 如果 edyn为空，说明内存分配失败，return 0。

7. 遍历32位动态节的每个外部表示：

动态节条目数加1。

如果外部表示的标签为 DT\_NULL，表示动态节结束，跳出循环。

8. 分配内存，用于存储32位动态节的内部表示，大小为动态节条目数乘以 Elf32\_Dyn结构的大小。

9. 如果动态节内部表示的内存分配失败：

打印内存不足的提示信息。

释放 edyn的内存。

返回0。

10. 遍历动态节的每个条目，进行内外部表示的转换：

将外部表示的标签转换为内部表示。

将外部表示的值转换为内部表示。

11. 释放 edyn的内存。

12. 返回1，表示成功获取32位动态节。

# 日期：2023年6月25日

**问题提出与分析**：

今天是实训的第六天，下午就要进行结项答辩了，基于前一段时间完成的任务，上午我们进行了材料的整理和整合，完成了答辩PPT和个人报告，总结了需要提交的内容，在上午我们团队成员分别进行了相互提问以应对下午答辩时老师可能提出的问题。

**组内讨论记录**：

我们分别设计了一些问题来进行模拟答辩

我提到的问题有：

1. Magic数的作用是什么？
2. 符号表的作用有哪些？
3. 文件头中程序入口地址表示的含义是什么？
4. 如何获取程序编译后物理地址内的信息，并分析其中的含义？

这些问题在与小组进行模拟答辩时我们进行了模拟提问。

**讨论总结**（包括但不仅限于问题解决方案，如分工情况、算法、系统设计方案等）：

通过这次相互提问的活动，我们不仅进一步加深了对项目的理解，还提高了团队的协作和沟通能力。我们彼此鼓励和支持，共同成长。我们相信这样的准备将为下午的答辩提供了充分的准备，让我们能够自信地回答老师的问题，并展示出我们项目的优势和创新之处。

我通过编写项目总结和报告的过程，通过进行PPT答辩以及撰写报告，进一步强化了自己的沟通和表达能力。通过将项目的设计和实现过程以及取得的成果进行详细的描述和总结，我不仅可以向他人清晰地展示我的工作，还可以反思和总结自己的经验和教训，以便在今后的项目中改进和提升。

下午我们成功完成了小组答辩，我们拟整理完所有材料进行itc系统的提交。

**实验过程记录**（包括但不仅限于实验方案设计、实验结果记录等）：

今日主要是进行了PPT和答辩内容以及个人报告的制作和撰写

文本

描述已自动生成

图 14 答辩PPT截图

图形用户界面, 应用程序

描述已自动生成

图 15 团队总汇报PPT共78页，包括详细解析

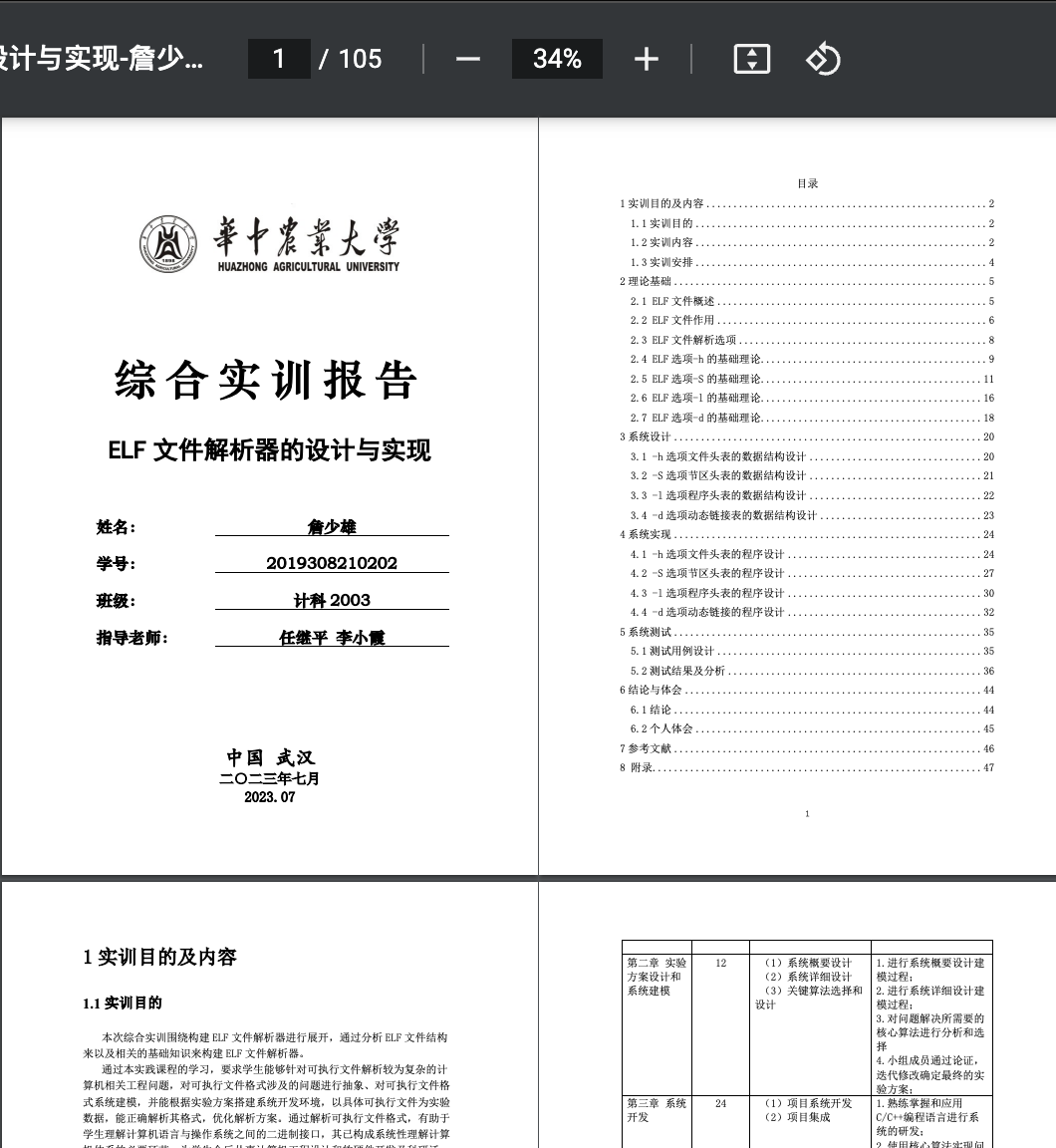


图 16 完整的个人报告

**致谢：**

最后感想团队成员的共同努力，没有大家的共同努力我，我们很难完成如何巨大的一个任务

同时也感谢任老师和李老师的指导和陪伴！

老师辛苦了！