

兰州大学学生创新创业行动计划

项目结项报告书

项目名称: ELF Database System(Elfdfs)

作者单位: 信息科学与工程学院

指导教师: 周庆国

类别:

☒ 自然科学类

☐ 社会科学类

共青团兰州大学委员会

1. 作者资料登记表

申报者情况	项目名称	ELF Database System(Elfdfs)		
	项目负责人	孙建蛟	所属学院	信息科学与工程学院
	专 业	计算机科学与技术	年 级	2006
	联系电话及邮件	Tel:13659499671 Email:sunjianjiao@gmail.com		
	指导教师	周庆国	所在学院	信息科学与工程学院
其他合作者情况	姓 名	学 院	专 业	年 级
	郭林丽	信息科学与工程学院	计算机科学与技术专业	2006
	谭龙	资源与环境学院	水文专业	2007

项目摘要	<p>打造出一个 ELF 文件的存储、查询统计、裁减和优化的平台,甚至为下一代可执行文件格式的开发提供依据。</p> <p>互联网(Internet)已经进入了 Web 2.0 的时代, Elfdbbs 充分利用“互联网这个平台”。通过 web 接口, 用户通过浏览器, 无需安装就可以使用 Elfdbbs。</p> <p>最终实现:</p> <ul style="list-style-type: none"> • 嵌入式系统需要的 ELF 可执行文件的裁减平台 • 下一代可执行文件格式的研发平台 • 程序开发人员的技术理论的实践平台 • 计算机病毒技术研究平台 • 基于 web2.0,不用安装, 就可以使用
学院创新创业领导小组意见	<p style="text-align: right;">组长:</p> <p style="text-align: right;">公章 (团委代章)</p>

注: 联系电话包括寝室电话、手机或其他可能的联系方式。

二. 结项作品情况 (自然科学类)

说明：1、必须由申请者本人填写；

2、作品分类请按作品的学术方向或所涉及的主要学科领域填写。

申请项目全称	ELF Database System
作品分类	(B) A. 机械与控制（包括机械、仪器仪表、自动化控制、工程、交通、建筑等） B. 信息技术（包括计算机、电信、通讯、电子等） C. 数理（包括数学、物理、地球与空间科学等） D. 生命科学（包括生物、农学、药学、医学、健康、卫生、食品等） E. 能源化工（包括能源、材料、石油、化学、化工、生态、环保等）
项目的目的和基本思路	立项目的： <ul style="list-style-type: none">• 互联网(Internet)已经进入了Web 2.0的时代，Elfdfs充分利用“互联网这个平台”。通过web接口，用户通过浏览器，无需安装就可以使用Elfdfs。• ELF是很多操作系统广泛支持的可执行文件格式，GNU工程的编译器GCC的最新版本更是对ELF提供全面支持。• 打造出一个ELF文件的存储、查询统计、裁减和优化的平台，甚至为下一代可执行文件格式的开发提供依据。

	<p>基本思路：</p> <p>1) 项目调研及准备 包括制定初步的项目计划；组建项目组；熟悉 ELF 格式标准、数据库等相关技术等。</p> <p>2) 项目设计 包括数据库设计；系统总体设计以及各子功能模块设计（如查询统计、裁减、优化）；系统测试方案设计等</p> <p>3) 项目实施 包括数据库的创建和系统各模块的编码实现。计划先实现具有基本功能的原型系统，然后逐步完善和改进。</p> <p>4) 系统测试 包括模块测试和整体测试，包括功能测试和性能测试等。</p> <p>5) 项目维护 项目最终以开放源代码的方式发布到兰大开源社区，让广大用户自由使用、报告 bug 并由项目组成员进行相关维护。</p>
项目的科学性、先进性及独特之处	<p>科学性：</p> <p>(1) 互联网(Internet)已经进入了 Web 2.0 的时代，网络普及使得人们可以随时随地连接到 Elfdb 的服务器。</p>

(2) ELF 存入数据库可行性分析

ELF 文件的格式：

Linking View	Eecution View																	
<table><tr><td>ELF header</td></tr><tr><td>Program header table</td></tr><tr><td><i>Optional</i></td></tr><tr><td>Section 1</td></tr><tr><td>.....</td></tr><tr><td>Section <i>n</i></td></tr><tr><td>.....</td></tr><tr><td>.....</td></tr><tr><td>Section header table</td></tr></table>	ELF header	Program header table	<i>Optional</i>	Section 1	Section <i>n</i>	Section header table	<table><tr><td>ELF header</td></tr><tr><td>Program header table</td></tr><tr><td>Segment 1</td></tr><tr><td>Segment 2</td></tr><tr><td>.....</td></tr><tr><td>Section header table</td></tr><tr><td><i>Optional</i></td></tr></table>	ELF header	Program header table	Segment 1	Segment 2	Section header table	<i>Optional</i>	可见， ELF 文件的 相关 数据结 构具有 关系表 的特征， 能够方 便地存 储到关 系数据 库系统 (如
ELF header																		
Program header table																		
<i>Optional</i>																		
Section 1																		
.....																		
Section <i>n</i>																		
.....																		
.....																		
Section header table																		
ELF header																		
Program header table																		
Segment 1																		
Segment 2																		
.....																		
Section header table																		
<i>Optional</i>																		

MySQL) 中。

(3) ELF 文件可小型化的分析

ELF 可执行文件有许多节区在链接时是没有用的，
例如：从 .comment 之后 Addr 的值都为 0，所以 .comment
之后的内容都是可以裁减掉的。

先进性及独特之处：

到目前为止，还没有发现一个这样的系统。

项目的实际应用价值和现实意义	<ul style="list-style-type: none"> • 嵌入式系统需要的 ELF 可执行文件的裁减平台 • 下一代可执行文件格式的研发平台 • 程序开发人员的技术理论的实践平台 • 计算机病毒技术研究平台 • 基于 web2.0，不用安装，就可以使用。
指导教师意见	<p>该项目组认真负责，到目前为止，已经基本完成项目的预期功能。</p> <p>在研究过程中，遇到问题，既能独立思考解决方案，又能主动与指导老师，研究生师兄学姐交流。同时该项目组分工明确，注重团队合作。</p>

三、作品打印处

一 引言

1. 相对于其它文件类型，可执行文件是一个操作系统中最重要文件类型，它们是完成操作的真正执行者。可执行文件的大小、运行速度、资源占用情况以及可扩展性、可移植性等与文件格式的定义和文件加载过程紧密相关。

2. 研究可执行文件的格式对编写高性能程序和一些黑客技术的运用都是非常有意义的。

3. 为什么针对 ELF

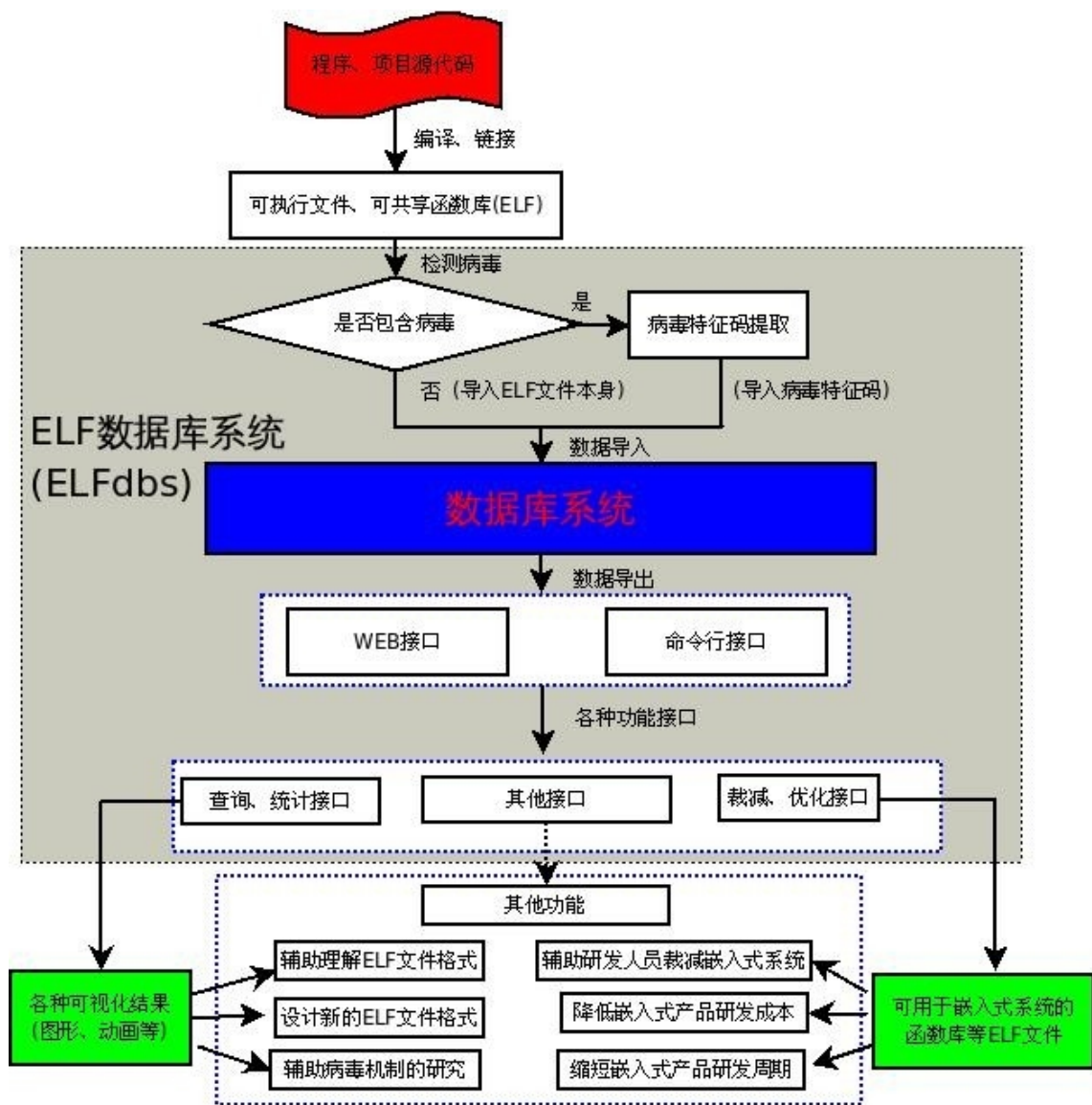
ELF 是很多平台下广泛支持的可执行文件格式，GNU 工程的编译器 gcc 最新版本更是对 elf 全力支持。

4. 嵌入式 Linux 一般是指对标准 Linux 发行版本进行小型化裁剪处理之后，适合于特定嵌入式应用场合的专用 Linux 操作系统。嵌入式系统通常是资源受限的系统，无论是处理器计算能力还是 RAM 或其他存储器容量都比较“小”。因此，如何创建一个小型化的可执行文件作为开发成为首先需要考虑的问题。

5. 计算机病毒的传播和感染，造成了极大的财产损失，甚至危急生命。

ELF Database System 正是为了解决上述问题而开发。

二 Elfdb 基本结构介绍



Elfdb 结构图 (图一)

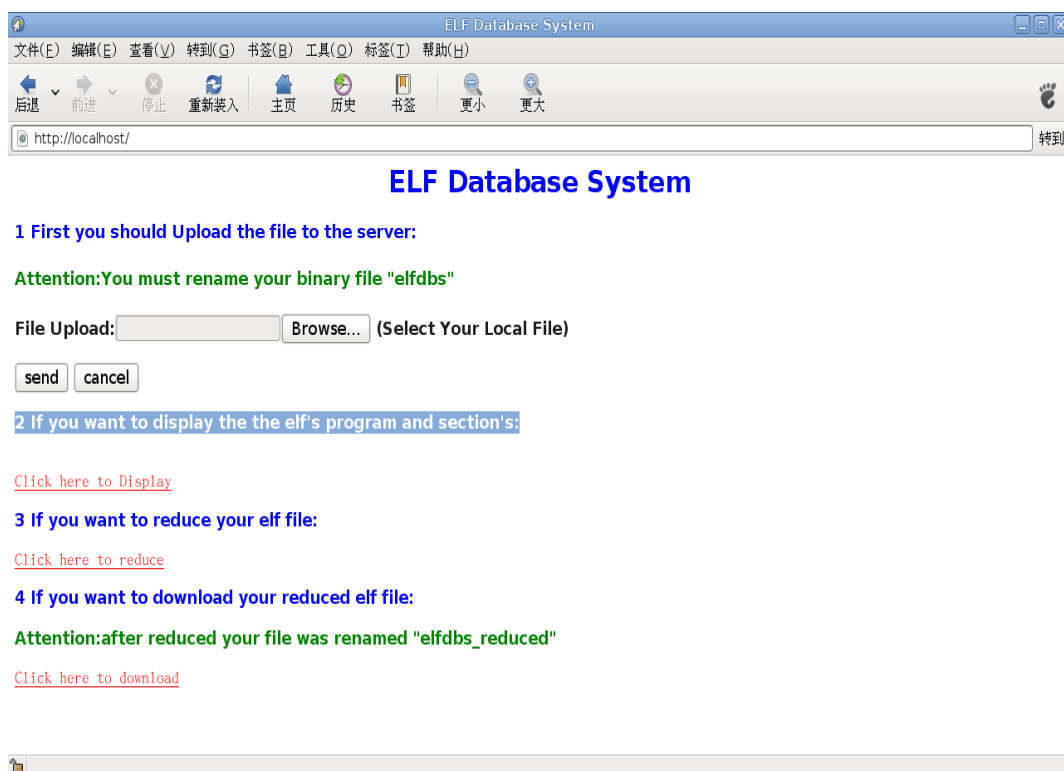
elfdbs使用流程图：



Elfdb使用流程图（图二）

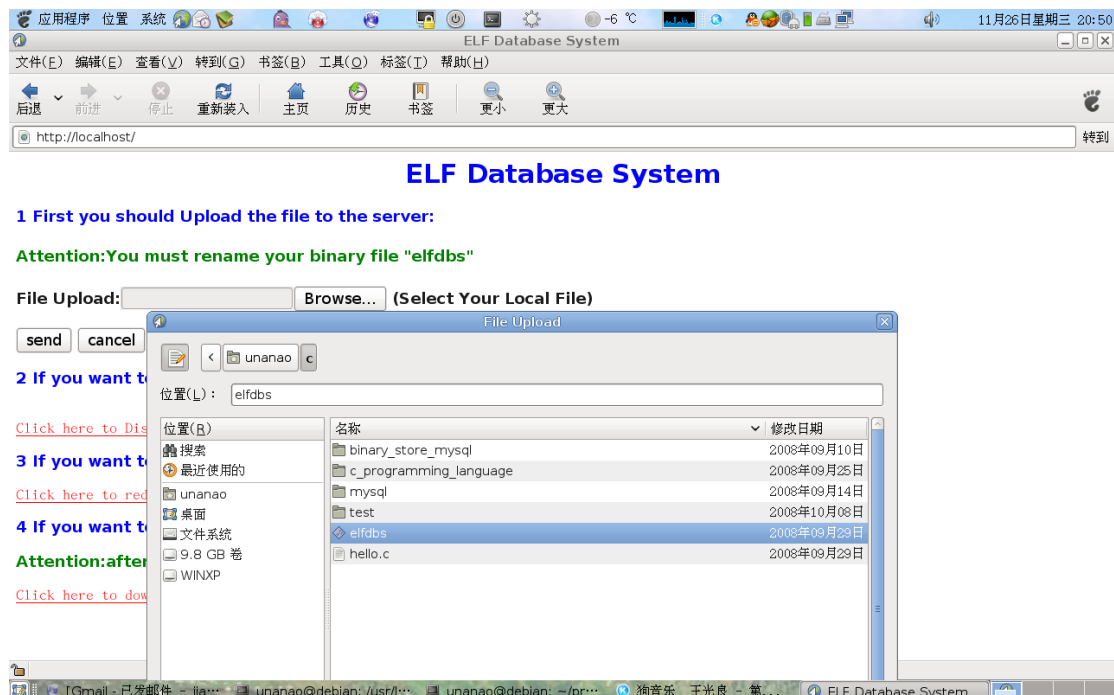
三 web 接口 -- 方便使用

3.1 通过 cgi 编程实现 web 接口：



Elfdb主页面(图三)

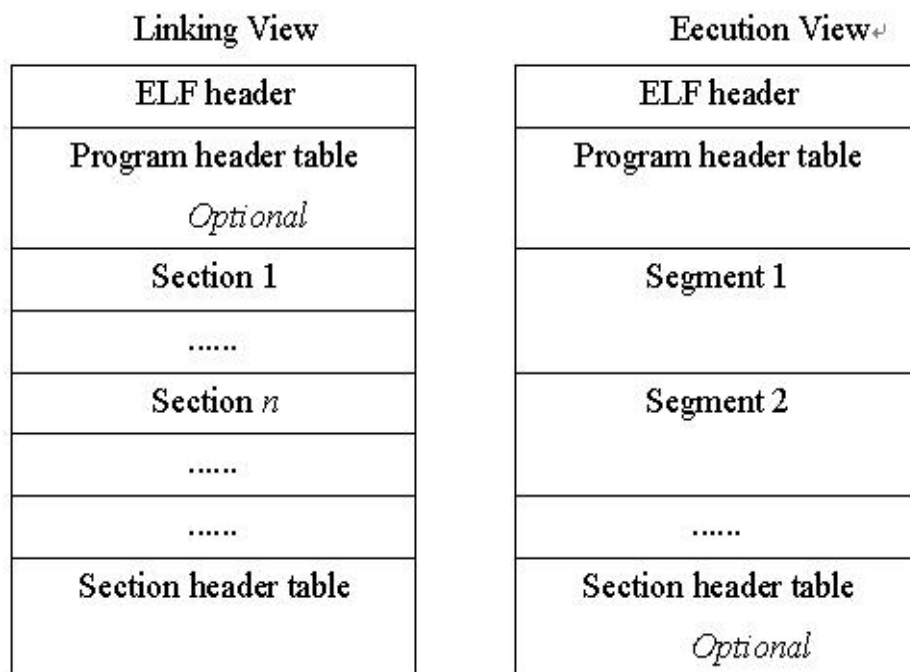
3.2 通 web 接口上传 Elf 文件到 Elfdb 服务器



Elfdb通过web接口上传ELF文件（图四）

四 ELF 文件分析及各部分的读取

ELF 文件格式：



4.1 读取 ELF header table:

文件开始处是一个 ELF 头部(ELF Header),用来描述整个文件的组织。节区部分包含链接视图的大量信息:指令、数据、符号表、重定位信息等等。

ELF header table 数据结构:

```
typedef struct
{
    unsigned char e_ident[EI_NIDENT];    /* Magic number
and other info */

    Elf32_Half    e_type;                  /* Object file
type */

    Elf32_Half    e_machine;               /* Architecture
*/

    Elf32_Word    e_version;              /* Object file
version */

    Elf32_Addr    e_entry;                /* Entry point
virtual address */

    Elf32_Off     e_phoff;                 /* Program
header table file offset */

    Elf32_Off     e_shoff;                /* Section
```

```

header table file offset */

    Elf32_Word    e_flags;                /* Processor-
specific flags */

    Elf32_Half    e_ehsize;              /* ELF header
size in bytes */

    Elf32_Half    e_phentsize;           /* Program
header table entry size */

    Elf32_Half    e_phnum;               /* Program
header table entry count */

    Elf32_Half    e_shentsize;          /* Section
header table entry size */

    Elf32_Half    e_shnum;               /* Section
header table entry count */

    Elf32_Half    e_shstrndx;           /* Section
header string table index */
} Elf32_Ehdr;

```

读取方法：

- (1) `lseek(fd,0,SEEK_SET);`
//找到首地址，定位到文件开始
- (2) `size=sizeof(Elf32_Ehdr);`

//确定 ELF header table 的大小

(3) read(fd,buf,size);

//读入 buf 中

4.2 读取 program header table

program header table 数据结构:

```
typedef struct {  
    Elf32_Word p_type;  
    Elf32_Off  p_offset;  
    Elf32_Addr p_vaddr;  
    Elf32_Addr p_paddr;  
    Elf32_Word p_filesz;  
    Elf32_Word p_memsz;  
    Elf32_Word p_flags;  
    Elf32_Word p_align;  
} Elf32_phdr;
```

可执行目标文件在 ELF 头部的 e_phentsize 和 e_phnum 成员中给出其自身程序头部 的大小, e_phoff 成员给出了程序自身头部的偏移。

读取 program header table 的 方法:

(1) lseek(fd,Elf32_Ehdr->e_phoff,SEEK_SET)

//通过 ELF 文件的头部找到 program header table 在文件中的位置

(2) size=Elf32_e_phentsize*Elf32_e_phnum;

//通过 ELF 头部找到 program header table 的大小
e_phnum 和表项数 // 目 e_phentsize，相乘得到
program header table 的大小

(3) read(fd,buf,size);

//将 program header table 读入 buf 中

4.3 读取 section header table

section header table 的数据结构：

```
typedef struct{  
    Elf32_Word sh_name;  
    Elf32_Word sh_type;  
    Elf32_Word sh_flags;  
    Elf32_Addr sh_addr;  
    Elf32_Off sh_offset;  
    Elf32_Word sh_size;  
    Elf32_Word sh_link;  
    Elf32_Word sh_info;  
    Elf32_Word sh_addralign;
```

```
Elf32_Word sh_entsize;  
}Elf32_Shdr;
```

可执行目标文件在 ELF 头部的 `e_shentsize` 和 `e_shnum` 成员中给出其自身程序头部的大小，`e_shoff` 成员给出了程序自身头部的偏移。

读取 section header table 的方法：

```
(1) lseek(fd,Elf32_Ehdr->e_shoff,SEEK_SET)
```

//通过 ELF 文件的头部找到 program header table 在文件中的位置

```
(2) size=Elf32_e_shentsize*Elf32_e_shnum;
```

//通过 ELF 头部找到 section header table 的大小
`e_shnum` 和表项数 // 乘 `e_shentsize`，相乘得到
section header table 的大小

```
(3) read(fd,buf,size);
```

//将 program header table 读入 buf 中

4.4 读取 ELF Section

分 2 步读取 ELF 的 section 资讯：

4.4.1 如图四，开始先读取 section header table 的资讯。

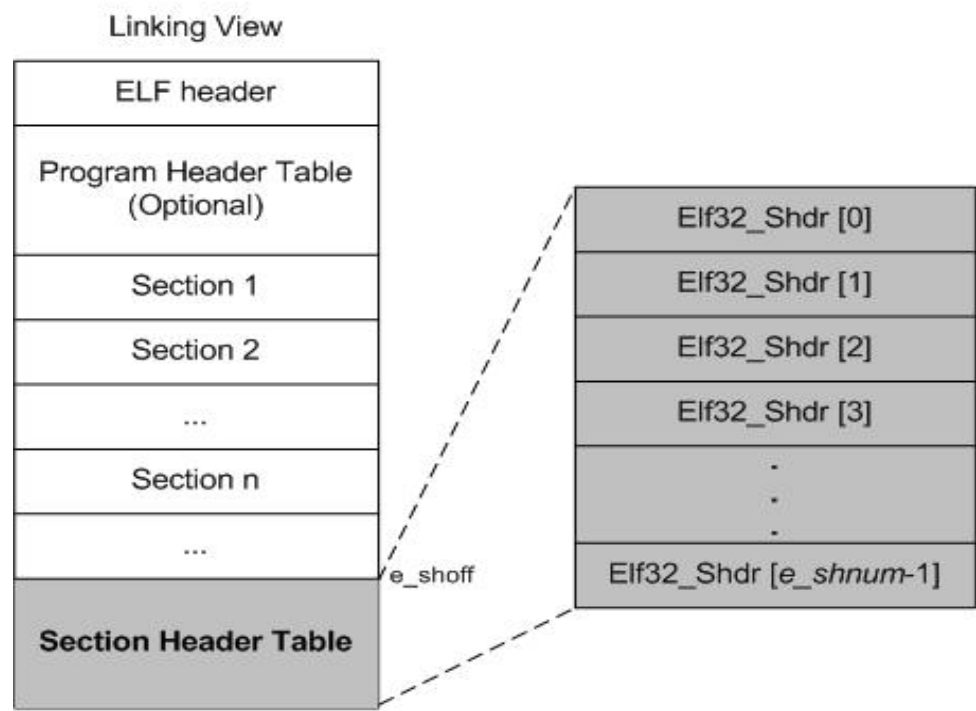
Section header table 是所有 section 的记录表格，所有的

section 都要透过 section header table 才能得知其在文件中的偏移位置 (offset)，如此一来才能读取 section 的内容。

4.4.2 如图五，接着再根据 section header table 读取文件的每一个 section。

section header table 里的 section 个数 (section entries) 记录于 ELF header 里的 *e_shnum*，每个 section entry 的长度则是记录在 ELF header 的 *e_shentsize* 中，单位是 bytes。

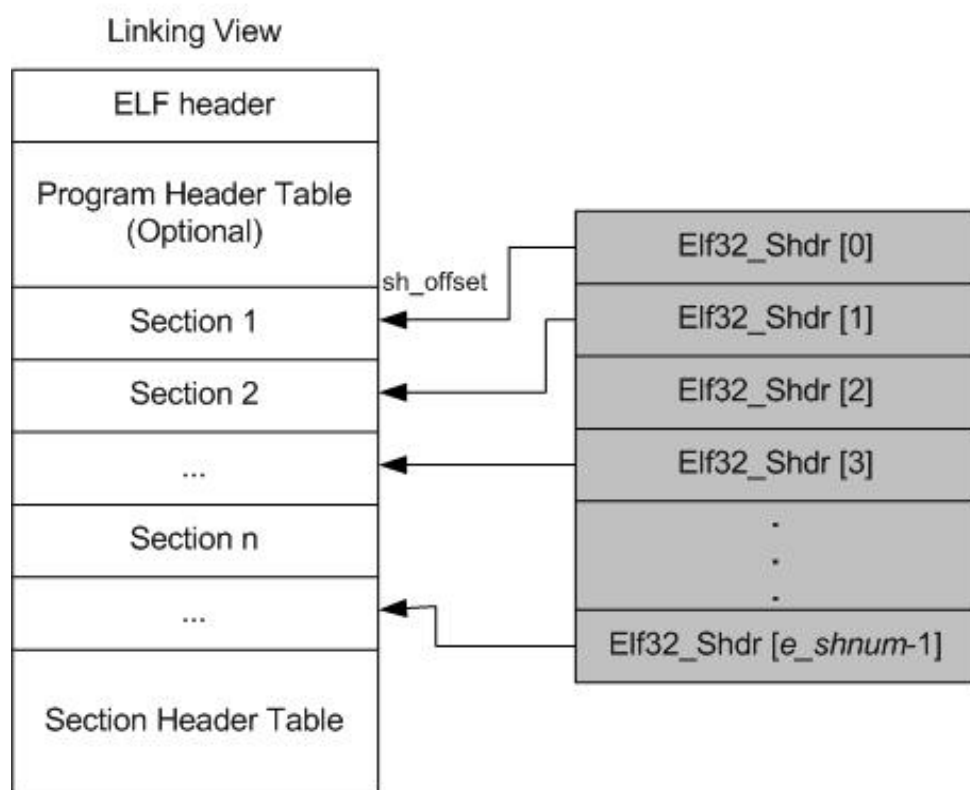
section header table:



section header table(图六)

ELF header 里的 *e_shoff*，是 section header table 开始的文件偏移位置（file offset）。因此，我们只要由 *e_shoff* 偏移值开始读取 *e_shnum* 个单位，每个单位为 *e_shentsize*（bytes），即可将整个 section header table 读取出来。

Section Entries:



section entries(图七)

SysV ABI 定义 section entry 的资料结构如下：

```
typedef struct {  
    Elf32_Word    sh_name;  
    Elf32_Word    sh_type;  
    Elf32_Word    sh_flags;  
    Elf32_Addr    sh_addr;  
    Elf32_Off     sh_offset;  
    Elf32_Word    sh_size;  
    Elf32_Word    sh_link;  
    Elf32_Word    sh_info;  
    Elf32_Word    sh_addralign;  
    Elf32_Word    sh_entsize;  
} Elf32_Shdr;
```

Section header table 是 *Elf32_Shdr* data type 的阵列，其元素个数为 *e_shnum*，我们可以透过索引 *Elf32_Shdr* 阵列来读取所有的 section，如图四所示，读取 Section Header Table。

同时，需要定义 1 个变量：

```
Elf32_Shdr header[MAX];
```

//MAX 是 section 的最多的个数

header[] 用来存放由 section header table 所读取出来的所有 section entry，其类型为 Elf32_Shdr；

将读指针移到 e_shoff 的地方，准备开始读取 section header table：

```
lseek(fd, Elf32_Ehdr->e_shoff, SEEK_SET);
```

然后确定每个 section 的大小，每个 section 的偏移可以被 2 整除必须把每个 section 本来的偏移位置和大小也存入到数据库中，两个 section 之间有一些没用的信息的，但是这个信息是需要保留的，因为考虑到 cpu 访问内存的时候有个内存地址对齐的问题。

```
if(Elf32_Shdr->sh_entsize%2!=0)
```

```
    size=Elf32_Shdr->sh_entsize+1;
```

```
else
```

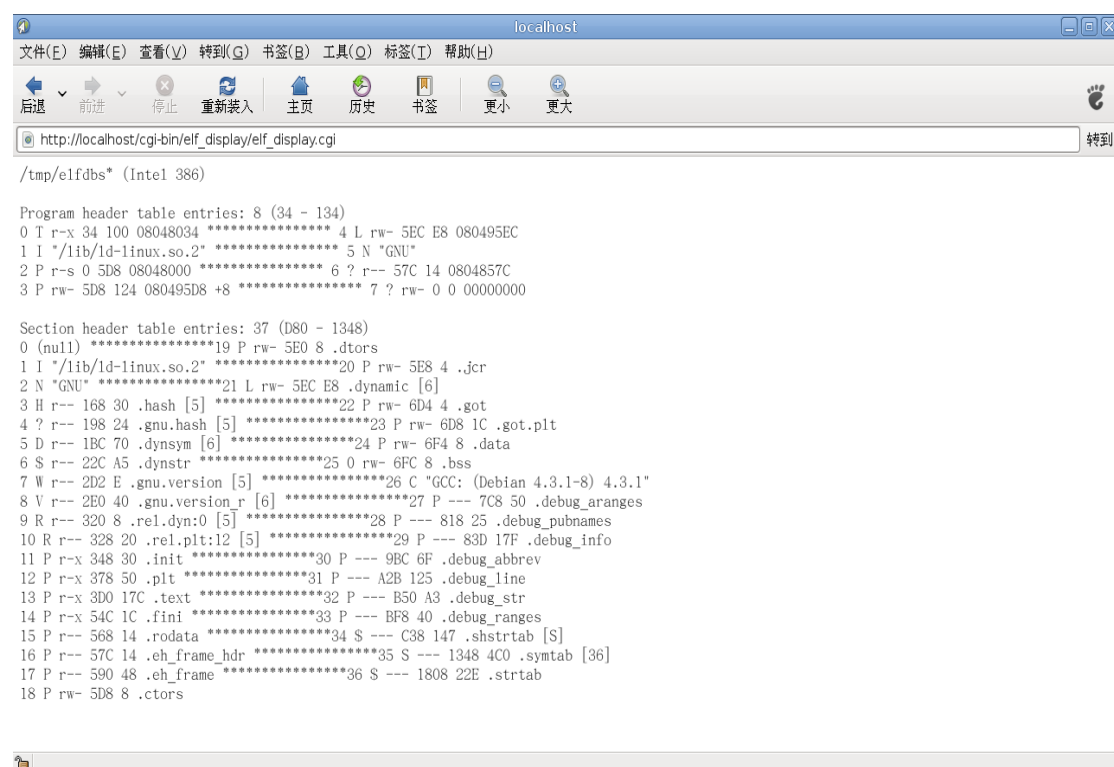
```
    size=Elf32_Shdr->sh_entsize;
```

//如果不能被 2 整除，需要加 1 操作

最后再利用最简单的方式，一次一个将所有的 section entry 读取出来。

```
for (i = 0; i < Elf32_Ehdr->e_shnum; i++) {  
    read(fd, &header[i], size);  
}
```

通过 web 接口分析 ELF 文件：



```
/tmp/elfdbs* (Intel 386)  
  
Program header table entries: 8 (34 - 134)  
0 T r-x 34 100 08048034 ***** 4 L rw- 5EC E8 080495EC  
1 I */lib/ld-linux.so.2 ***** 5 N *GNU*  
2 P r-s 0 5D8 08048000 ***** 6 ? r-- 57C 14 0804857C  
3 P rw- 5D8 124 080495D8 +8 ***** 7 ? rw- 0 0 00000000  
  
Section header table entries: 37 (D80 - 1348)  
0 (null) *****19 P rw- 5E0 8 .dtors  
1 I */lib/ld-linux.so.2 *****20 P rw- 5E8 4 .jcr  
2 N *GNU* *****21 L rw- 5EC E8 .dynamic [6]  
3 H r-- 168 30 .hash [5] *****22 P rw- 6D4 4 .got  
4 ? r-- 198 24 .gnu.hash [5] *****23 P rw- 6D8 1C .got.plt  
5 D r-- 1BC 70 .dynsym [6] *****24 P rw- 6F4 8 .data  
6 S r-- 22C A5 .dynstr *****25 0 rw- 6FC 8 .bss  
7 W r-- 2D2 E .gnu.version [5] *****26 C *GCC: (Debian 4.3.1-8) 4.3.1*  
8 V r-- 2E0 40 .gnu.version_r [6] *****27 P --- 7C8 50 .debug_aranges  
9 R r-- 320 8 .rel.dyn:0 [5] *****28 P --- 818 25 .debug_pubnames  
10 R r-- 328 20 .rel.plt:12 [5] *****29 P --- 83D 17F .debug_info  
11 P r-x 348 30 .init *****30 P --- 9BC 6F .debug_abbrev  
12 P r-x 378 50 .plt *****31 P --- A2B 125 .debug_line  
13 P r-x 3D0 17C .text *****32 P --- B50 A3 .debug_str  
14 P r-x 54C 1C .fini *****33 P --- BF8 40 .debug_ranges  
15 P r-- 568 14 .rodata *****34 S --- C38 147 .shstrtab [S]  
16 P r-- 57C 14 .eh_frame_hdr *****35 S --- 1348 4C0 .symtab [36]  
17 P r-- 590 48 .eh_frame *****36 S --- 1808 22E .strtab  
18 P rw- 5D8 8 .ctors
```

通过 web 接口显示 ELF 文件格式（图八）

五 如何将 ELF 二进制文件存入数据库

5.1 MySQL 有一个二进制数据类型 longblob，可以用来存放二进制数据。

```
CREATE table elf(id int,data longblob);
```

```
//为 ELF 的各个部分创建 table
```

5.2 通过 `mysql_real_escape_string` 函数在二进制数据外面加一层“壳”。

```
*end++='\'';
```

```
end+=mysql_real_escape_string(conn,end,buf,n);
```

```
*end++='\'';
```

```
*end++=')';
```

```
//转换 NUL(ASCII 0)、'\n'、'\r'、'\''、'\"'和 Control-Z  
//等等
```

六 对可执行文件的裁减

找出可以裁减掉的部分

```
unanao@debian:~/c$ readelf -S hello
```

There are 37 section headers, starting at offset 0xd80:

Section Headers:

[Nr]	Name	Type	Addr	Off
[0]		NULL		00000000
000000	000000 00	0 0 0		
[1]	.interp	PROGBITS		08048134

000134	000013	00	A	0	0	1	
[2]	.note.ABI-tag						NOTE 08048148
000148	000020	00	A	0	0	4	
[3]	.hash						HASH 08048168
000168	000030	04	A	5	0	4	
[4]	.gnu.hash						GNU_HASH 08048198
000198	000024	04	A	5	0	4	
[5]	.dynsym						DYNSYM 080481bc
0001bc	000070	10	A	6	1	4	
[6]	.dynstr						STRTAB 0804822c
00022c	0000a5	00	A	0	0	1	
[7]	.gnu.version						VERSYM 080482d2
0002d2	00000e	02	A	5	0	2	
[8]	.gnu.version_r						VERNEED 080482e0
0002e0	000040	00	A	6	2	4	
[9]	.rel.dyn						REL 08048320
000320	000008	08	A	5	0	4	
[10]	.rel.plt						REL 08048328
000328	000020	08	A	5	12	4	
[11]	.init						PROGBITS 08048348
000348	000030	00	AX	0	0	4	

[12]	.plt		PROGBITS	08048378
000378	000050	04	AX 0 0 4	
[13]	.text		PROGBITS	080483d0
0003d0	00017c	00	AX 0 0 16	
[14]	.fini		PROGBITS	0804854c
00054c	00001c	00	AX 0 0 4	
[15]	.rodata		PROGBITS	08048568
000568	000014	00	A 0 0 4	
[16]	.eh_frame_hdr		PROGBITS	0804857c
00057c	000014	00	A 0 0 4	
[17]	.eh_frame		PROGBITS	08048590
000590	000048	00	A 0 0 4	
[18]	.ctors		PROGBITS	080495d8
0005d8	000008	00	WA 0 0 4	
[19]	.dtors		PROGBITS	080495e0
0005e0	000008	00	WA 0 0 4	
[20]	.jcr		PROGBITS	080495e8
0005e8	000004	00	WA 0 0 4	
[21]	.dynamic		DYNAMIC	080495ec
0005ec	0000e8	08	WA 6 0 4	
[22]	.got		PROGBITS	080496d4

0006d4	000004	04	WA	0	0	4	
[23]	.got.plt						PROGBITS 080496d8
0006d8	00001c	04	WA	0	0	4	
[24]	.data						PROGBITS 080496f4
0006f4	000008	00	WA	0	0	4	
[25]	.bss						NOBITS 080496fc
0006fc	000008	00	WA	0	0	4	
[26]	.comment						PROGBITS 00000000
0006fc	0000cb	00		0	0	1	
[27]	.debug_aranges						PROGBITS 00000000
0007c8	000050	00		0	0	8	
[28]	.debug_pubnames						PROGBITS 00000000
000818	000025	00		0	0	1	
[29]	.debug_info						PROGBITS 00000000
00083d	00017f	00		0	0	1	
[30]	.debug_abbrev						PROGBITS 00000000
0009bc	00006f	00		0	0	1	
[31]	.debug_line						PROGBITS 00000000
000a2b	000125	00		0	0	1	
[32]	.debug_str						PROGBITS 00000000
000b50	0000a3	01	MS	0	0	1	

[33]	.debug_ranges	PROGBITS	00000000
000bf8	000040 00	0 0 8	
[34]	.shstrtab	STRTAB	00000000
000c38	000147 00	0 0 1	
[35]	.symtab	SYMTAB	00000000
001348	0004c0 10	36 55 4	
[36]	.strtab	STRTAB	00000000
001808	00022e 00	0 0 1	

Key to Flags:

W (write), A (alloc), X (execute), M (merge), S
(strings)

I (info), L (link order), G (group), x (unknown)

0 (extra OS processing required) o (OS specific), p
(processor specific)

从.comment 之后 Addr 的值都为 0，所以.comment 之后的内容都是可以裁减掉的。

然后将各个段按照存入数据库的地址，写入文件，就得到裁减后的二进制文件。

七 裁减后文件的下载

搭建配置 vsftp，通过 vsftp 匿名下载：

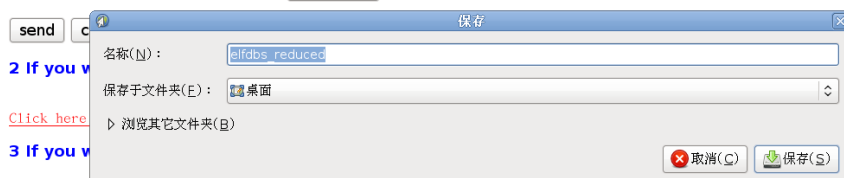


ELF Database System

1 First you should Upload the file to the server:

Attention: You must rename your binary file "elfdbs"

File Upload: /home/unanao/c/elfdb: Browse... (Select Your Local File)



2 If you w

[Click here](#)

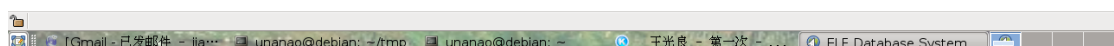
3 If you w

[Click here to view](#)

4 If you want to download your reduced elf file:

Attention: after reduced your file was renamed "elfdbs_reduced"

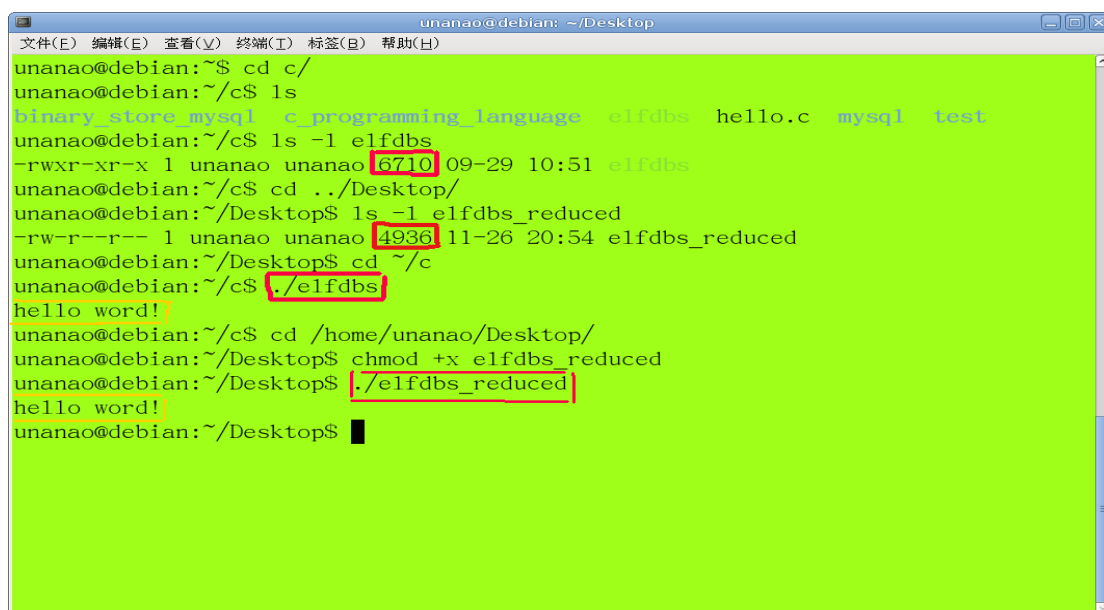
[Click here to download](#)



通过 web 接口直接下载裁减后的文件（图九）

八 裁减前后对照

裁减前的二进制文件 elfdbs 大小为 6710Byte，裁减后的文件 elfdbs_reduced 大小为 4936Byte 小了近 2 KB，依然可以执行



九 总结

近年来嵌入式 Linux 技术迅速发展,各种商业和开放源码的 Linux 发行版为不同的硬件平台、不同的应用环境提供了多种选择。Linux 已经越来越广泛地应用于各种嵌入式设备中。但是嵌入式设备的存储空间相对较小。所以我们如何得到更精简的可执行文件是我们不断努力的方向,与此同时研究可执行文件的格式对编写高性能程序有重要意义。所以 Elfdb 正是为了解决这些问题而开发。