ELF Files

The C Code: main.c

```
int fib[10], n = 4, arg[10] = {5, 7, 10, 13} ;
int main()
{
 int fibonacci(int), i ;

 for(i=0; i<n; ++i) {
     fib[i] = fibonacci(arg[i]);
      printf("fib(%d) = %d\n", arg[i], fib[i]);
 }
}
```

Object Module: main.o

```
$ cc -c main.c
$ ls -l main.c main.o
-rw-r--r-- 1 goutam users  194 Jul 30 16:25 main.c
-rw-r--r-- 1 goutam users 1108 Aug  9 16:42 main.o
$ file main.c main.o
main.c: ASCII text
main.o: ELF 32-bit LSB relocatable, Intel 80386,
        version 1 (SYSV), not stripped
$
```

What is there in main.o?

Inside main.o

nm list symbols from object files

```
$ nm -A main.o
main.o:00000020 D arg
main.o:00000028 C fib
main.o:         U fibonacci
main.o:00000000 T main
main.o:00000000 D n
main.o:         U printf
$
```

Global Symbols of main.o

Symb.	Val.	Meaning
arg	0x20 (address)	Init.data (D-.data)
fib	0x28 (size)	UnInit.data (C-.common)
fibonacci		Undefined (U)
main	0x0 (address)	Code (T - .text)
n	0x0 (address)	Init. data (D - .data)
printf		Undefined (U)

Inside main.o

objdump **display information from object files.**

```
$ objdump -d main.o
main.o:     file format elf32-i386
Disassembly of section .text:
00000000 <main>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
        .........................
  72:   c9                      leave
  73:   c3                      ret
```

Inside main.o

readelf **Displays information about ELF files.**

```
$ readelf -r main.o
Relocation section '.rel.text' at offset 0x414 contains 8 entries:
 Offset     Info    Type          Sym.Value  Sym. Name
0000001d  00000701 R_386_32     00000000   n
00000031  00000801 R_386_32     00000020   arg
00000036  00000a02 R_386_PC32   00000000   fibonacci
00000040  00000b01 R_386_32     00000020   fib
0000004d  00000b01 R_386_32     00000020   fib
00000057  00000801 R_386_32     00000020   arg
0000005c  00000501 R_386_32     00000000   .rodata
00000061  00000c02 R_386_PC32   00000000   printf
```

How does nm, objdump and readelf

Extract Information from ELF File?

ELF Format

Object Module Format

ELF Header
Section$_0$
. . .
Section$_k$
Section Header Table (SHT)

ELF Header

```
typedef struct {
        unsigned char   e_ident[16];
        Elf32_Half      e_type;
        Elf32_Half      e_machine;
        Elf32_Word      e_version;
        Elf32_Addr      e_entry;
        Elf32_Off       e_phoff;
        Elf32_Off       e_shoff;
```

ELF Header

```
        Elf32_Word      e_flags;
        Elf32_Half      e_ehsize;
        Elf32_Half      e_phentsize;
        Elf32_Half      e_phnum;
        Elf32_Half      e_shentsize;
        Elf32_Half      e_shnum;
        Elf32_Half      e_shstrndx;
} Elf32_Ehdr;
```

Header size is 52B

A C Program for Byte Dump

```
#include <stdio.h>
int main() {
 int c, index = 0;

 while((c = getchar()) != EOF)
   printf(" %d.  %d  0x%X\n",index++,c,c);
}
```

Byte Dump main.o

```
0.  127  0x7F
1.  69  0x45
2.  76  0x4C
3.  70  0x46
4.  1  0x1
  ..............
1104.  2  0x2
1105.  12  0xC
1106.  0  0x0
1107.  0  0x0
```

Header of `main.o`

Offset (D)	Code (H)	Interpretation
0	7F 45 4C 46	"0x7F ELF"
4	01	32-bit Object
5	01	Little-endian
6	01	Version
7	00 $\cdots$ 00	Zeros

This part of the header is system independent.

Each data is of size 1B.

Header of `main.o`

Offset(D)	Code (H)	Interpretation
16	01 00	Relocatble file
18	03 00	Intel386
20	01 00 00 00	Version
32	8C 01 00 00	Offset SHT 0x18C=396
40	34 00 00 00	Size - ELF Header
46	28 00	Size - SHT entry
48	0A 00	No. of Entries - SHT
50	07 00	SHT index - Str. Tab.

Section Header Table: 0th Entry

```
396.  0x0
397.  0x0
   ..................
434.  0x0
435.  0x0
```

All entries are zero (0).

Section Header Table

```
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: 7th Entry

Offset (D)	Code (H)	Interpretation
676	11 00 00 00	Section name (index)
		.shstrtab
680	03 00 00 00	Header of String Tab.
692	49 01 00 00	Section offset in file
		0x149 = 329
696	41 00 00 00	Size of the section
708	01 00 00 00	No alignment
712	00 00 00 00	

String Table for Section Names: Offset 329B

```
329: 0 46 115 121 109 116 97 98 0
338: ..............
385: 46 99 111 109 109 101 110 116 0
```

Offset (D)	Code (H)	Interpretation
329	00	
330	2E 73 ··· 62 00	".symtab"
338	2E 73 ··· 62 00	".strtab"
346	2E 73 ··· 62 00	".shstrtab"
356	2E 72 ··· 74 00	".rel.text"
366	2E 64 ··· 61 00	".data"
372	2E 62 ··· 73 00	".bss"
377	2E 72 ··· 61 00	".rodata"
385	2E 63 ··· 74 00	".comment"

Offset for .text

329:	\0									
356:	**46**	**114**	**101**	**108**	**46**	**116**	**101**	**120**	**116**	**0**
	·	**r**	**e**	**l**	·	**t**	**e**	**x**	**t**	\0

Offset for .text is $360 - 329 = 31 =$ **0x1F**

Search for SHT Entry for .text

- **Section Header Table (SHT) starts from the file offset 396.**
- **Size of each SHT entry is 40B.**
- **The first 4B of each entry gives the offset in the section name string table for the name of the section.**
- **For the .text section we search for the offset entry 0x1F.**

Section Header Table: 1st Entry

Offset (D)	Code (H)	Interpretation
436	1F 00 00 00	Section name (index)
		.text
440	01 00 00 00	Program Info.
444	06 00 00 00	Instr. + Occ. Mem.
452	34 00 00 00	File Offset (52B)
456	74 00 00 00	Size 116B

Machine Code : `main.o` : 116 Bytes

Starting Address : *Byte 52* of `main.o`

**55 89 E5 53 83 EC 4 83 E4 F0 B8 0 0 0 0 29 C4
C7 45 F8 0 0 0 0 8B 45 F8 3B 5 0 0 0 0 7C 2 EB
4A 8B 5D F8 83 EC C 8B 45 F8 FF 34 85 0 0 0 0
E8 FC FF FF FF 83 C4 10 89 4 9D 0 0 0 0 83 EC
4 8B 45 F8 FF 34 85 0 0 0 0 8B 45 F8 FF 34 85 0
0 0 0 68 0 0 0 0 E8 FC FF FF FF 83 C4 10 8D 45
F8 FF 0 EB A9 8B 5D FC C9 C3**

Machine Code : `main.o` : Locations to Relocate

Address	*M/c Code*	*Assembly Code*
0	55	pushl %ebp
1	89 E5	movel %esp, %ebp
...	...	...
18 **L2:**	8B 45 F8	
1B	3B 05	cmpl n, %eax
	00 00 00 00	
21	7C 02	jl L5 # (23+2 = 25)
23	EB 4A	jmp L3 # (25+4A = 6F)
25 **L5:**	8B 5D F8	...

Address	*M/c Code*	*Assembly Code*
2E	FF 34 85	pushl arg(,%eax,4)
	$\boxed{\text{00 00 00 00}}$	
35	E8	call fibonacci
	$\boxed{\text{FC FF FF FF}}$	
...	...	...
3D	89 04 9D	movl %eax, fib(,%ebx,4)
	$\boxed{\text{00 00 00 00}}$	

Address	*M/c Code*	*Assembly Code*
4A	FF 34 85	pushl fib(,%eax,4)
	$\boxed{\text{00 00 00 00}}$	
...	...	...
54	FF 34 85	pushl arg(,%eax,4)
	$\boxed{\text{00 00 00 00}}$	
5B	68	pushl $.LC0
	$\boxed{\text{00 00 00 00}}$	
60	E8	call printf
	$\boxed{\text{FC FF FF FF}}$	

Address	*M/c Code*	*Assembly Code*
6D	EB A9	jmp L2 # (6F + A9 = 18)
6F **L3:**	8B 5D FC	$\cdots$
72	C9	leave
73	C3	ret

Relocation Entries

- **There are eight (8) relocation entries in the code.**
- **This information is available in the section .rel.text of main.o.**

Offset for .rel.text

329:	\0									
356:	**46**	**114**	**101**	**108**	**46**	**116**	**101**	**120**	**116**	**0**
	·	**r**	**e**	**l**	·	**t**	**e**	**x**	**t**	\0

Offset for .rel.text is $356 - 329 = 27 =$ **0x1B**

Section Header Table: 2nd Entry

Offset (D)	Code (H)	Interpretation
476	1B 00 00 00	Section name (index) .rel.text
480	09 00 00 00	Reloc. Enries
492	14 04 00 00	File Offset (1044B)
496	40 00 00 00	Size 64B $= 8 \times$ 8B
504	01 00 00 00	SHT index of .text
512	08 00 00 00	Each ent. size

Data Type for Relocation Entry

```
typedef struct {
Elf32_Addr r_offset;
Elf32_Word r_info;
} Elf32_Rel;
```

The first 4Byte of a relocation entry gives the file offset (in the object file) of the point where the relocation is to be made. Each relocation is for an address and is 4Byte long.

Section .rel.text: Offset 329B

Section Offset	Reloc. Offset
1044	0x1D 0x0 0x0 0x0
1052	0x31 0x0 0x0 0x0
1060	0x36 0x0 0x0 0x0
1068	0x40 0x0 0x0 0x0
1076	0x4D 0x0 0x0 0x0
1084	0x57 0x0 0x0 0x0
1092	0x5C 0x0 0x0 0x0
1100	0x61 0x0 0x0 0x0

Assignment IV

Write a C program that will read an object file (*.o) and will print file offsets of all the relocation entries of the .text. You may also try to print the names of the objects for which the relocations are necessary (this information is available in the file).

Do not use any standard utility e.g. readelf, objdump, nm etc.

URLs

`http://www.caldera.com/developers/gabi/2003-12-17/contents.html`

`http://www.cs.ucdavis.edu/~haungs/paper/node10.html`