ELF1 7 Examples - 1 Introduction - ELF Study 1999

Young W. Lim

2019-12-10 Tue

Outline

Based on

"Study of ELF loading and relocs", 1999
http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Compling 32-bit program on 64-bit gcc

- `gcc -v`
- `gcc -m32 t.c`
- `sudo apt-get install gcc-multilib`
- `sudo apt-get install g++-multilib`
- `gcc-multilib`
- `g++-multilib`
- `gcc -m32`
- `objdump -m i386`

TOC: Relocs in Design Cycles

TOC: Relocation background

Symbol types

global Symbols *defined* by module m
- referenced by other modules
- *nonstatic* c functions
- *nonstatic* global variables
global Symbols *referenced* by module m
- defined by other module
- *external* c functions
- *external* global variables
local Symbols *defined* by module m
- referenced by module m exclusively
- *static* c functions
- *static* global variables

Struct Rel type

```
struct Rel {
  uint32 r_offset;          // the reloc target location
  uint24 r_sym_index;       // the symbol name
  uint8  r_type;            // the reloc algorithm
};
```

```
enum for t_type {
  R_386_32         =1,
  R_386_GOT32      =3,
  R_386_PLT32      =4,
  R_386_COPY       =5,
  R_386_GLOB_DAT   =6,
  R_386_JUMP_SLOT  =7,
  R_386_RELATIVE   =8,
  R_386_GOTOFF     =9,
  R_386_GOTPC      =10
};
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Reloc access

- when the reloc algorithm (r_type) is invoked,
  it has direct access to
  - the reloc target location (r_offset)
  - the symbol name (through r_sym_index)

  it has indirect access to
  - the data currently at the target location (addend)
  - the object length, through the symbol description

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Struct Rela type

- Some architectures, like M68k
  use a differentrelc, Rela
  which has one extra parameter (addend)
  this relocs 12 bytes each, instead of 8
- The Rel is just as flexible

`http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html`

Struct Rel v.s. Rela (modern) types

```
struct Rel {
  uint32 r_offset;          // the reloc target location
  uint24 r_sym_index;       // the symbol name
  uint8  r_type;            // the reloc algorithm
};

----------------------------

typedef struct {
  Elf32_Addr    r_offset;
  Elf32_Word    r_info;    // r_sym_index + r_type
} Elf32_Rel

typedef struct {
  Elf32_Addr    r_offset;
  Elf32_Word    r_info;    // r_sym_index + r_type
  Elf32_Sword   r_addend;
} Elf32_Rela
```

http://www.cs.cmu.edu/afs/cs/academic/class/15213-s01/s00/doc/elf.pdf

ELF Relocation Entry Types - Elf32_Rel, Elf32_Rela

Elf32_Rel

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

Elf32_Rela

```
typedef struct {
        Elf32_Addr      r_offset;
        Elf32_Word      r_info;
        Elf32_Sword     r_addend;
} Elf32_Rela
```

http://www.cs.cmu.edu/afs/cs/academic/class/15213-s01/s00/doc/elf.pdf

Relocation sections in /usr/bin/dir (1)

- readelf -r → .rel.got, .rel.bss

```
Relocation section '.rel.got' at offset 0xb6c contains 1 entries:
Offset   Info  Type              Symbol's Value Symbol's Name
08054748 00106 R_386_GLOB_DAT          00000000 __gmon_start__

Relocation section '.rel.bss' at offset 0xb74 contains 8 entries:
Offset   Info  Type          Symbol's Value Symbol's Name
08054800 04405 R_386_COPY 08054800       __ctype_tolower
08054804 00605 R_386_COPY 08054804       stdout
08054808 03505 R_386_COPY 08054808       stderr
0805480c 01905 R_386_COPY 0805480c       __ctype_toupper
08054810 01105 R_386_COPY 08054810       _nl_msg_cat_cntr
08054814 00905 R_386_COPY 08054814       __ctype_b
08054818 01405 R_386_COPY 08054818       optarg
0805481c 02205 R_386_COPY 0805481c       optind
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Relocation sections in /usr/bin/dir (2)

- readelf -r → .rel.plt

```
Relocation section '.rel.plt' at offset 0xbb4 contains 58 entries:
Offset    Info   Type            Symbol's Value Symbol's Name
08054660 00e07 R_386_JUMP_SLOT 08048dc4       readlink
08054664 03c07 R_386_JUMP_SLOT 08048dd4       getgrnam
08054668 02407 R_386_JUMP_SLOT 08048de4       ferror
0805466c 04107 R_386_JUMP_SLOT 08048df4       strchr
08054670 01007 R_386_JUMP_SLOT 08048e04       __overflow
08054674 04507 R_386_JUMP_SLOT 08048e14       __register_frame_info
08054678 01f07 R_386_JUMP_SLOT 08048e24       _obstack_begin
0805467c 02b07 R_386_JUMP_SLOT 08048e34       fnmatch
08054680 02907 R_386_JUMP_SLOT 08048e44       localtime
08054684 02f07 R_386_JUMP_SLOT 08048e54       strcmp
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Relocs in Design cycles

1. .o files for executbles
   R_386_PC32, R_386_32
2. .o files for shared libraries

	local symbols	global symbols
code	R_386_GOTOFF	R_386_GOT32, R_386_PLT32
data	R_386_32	R_386_32
	by the section number	by the symbol name

3. executables
   R_386_COPY, R_386_JMP_SLOT
4. shared libraries
   R_386_RELATIVE, R_386_GLOB_DAT, R_386_JMP_SLOT

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

PIC reloc summary in object (.o) files (1)

- R_386_GOT32 for global symbols in the code section
  - the relative distance of the GOT entry from GOT[0]
  - the linker will store a pointer to the given global symbol
  - used to indirectly reference a global symbol
- R_386_GOTOFF for local symbols in the code section
  - the relative distance of the given symbol from GOT[0]
  - the linker has placed a pointer to the given local symbol
  - used to address static data (a local symbol)

Linkers and Loaders, J. R. Levine

PIC reloc summary in object (.o) files (2)

- R_386_32 for global symbols in the data section
  - references the symbol by the name
- R_386_32 for local symbols in the data section
  - references the symbol by the section number (section-plus-offset)

`Linkers and Loaders, J. R. Levine`

PIC reloc summary in object (.o) files (3)

- `R_386_PLT32` for function symbols
    - the relative distance from the function call to the PLT entry
    - the linker will store a pointer to the corresponding GOT entry
    - GOT entry is used to indirectly reference a function symbol

`Linkers and Loaders, J. R. Levine`

Relocs in a PIC shared object (.so) file (1)

- R_386_GLOB_DAT for global symbols
    - used for an indirect reference to a global symbol in a PIC shared library
- R_386_RELATIVE for loading shared libraries
    - used to mark data address in a PIC shared library
    - that need to be relocated at load time
- R_386_JUMP_SLOT for function symbols
    - used for an indirect reference to a function symbol in a PIC shared library

`Linkers and Loaders, J. R. Levine`

Relocs in a PIC shared library (.so) file (2)

`R_386_JMP_SLOT`	S	• *PIC* reference to a function symbol • offset : a PLT entry location • fill the GOT entry with a function symbol address
`R_386_GLOB_DAT`	S	• *PIC* reference to a global symbol • offset to a GOT entry • fill the GOT entry with a global symbol address
`R_386_RELATIVE`	B+A	• *PIC* reference to a local symbol • offset to a section • add the load address to the relative address (offset)

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Relocs in a non-PIC executable file

R_386_COPY	None	• *non-PIC* reference to a global symbol • offset : a location in a WR segment • copy the library symbol data into an app's data space
R_386_JMP_SLOT	S	• *PIC* reference to a function symbol • offset : a PLT entry location • fill the GOT entry with a function symbol address

- R_386_GLOB_DAT : not used in a *non-PIC* executable file
- in the recently released linux, PIE is enforced by default
    - no difference in shared library relocs and executable relocs

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

PIC, PIE, and non-PIC executables

- by default position indepedent execution (PIE)
    - functions defined in the same module
      GOT is utilized, but PLT is not used
    - function references, which are externally defined
      both GOT and PLT are utilized
- to enforce the PIC features, use `-fPIC`
  then both GOT and PLT are used
- to disable the PIC feature use `-fno_pic`
  then neither GOT nor PLT is used
    - neither `R_386_PLT32` nor `R_386_GOT32` reloc is used
    - only `R_386_32` and `R_386_PC32` are used

Using GOT in a module

- to use GOT in a module
  the reference to GOT is necessary
  (`GLOBAL_OFFSET_TABLE`)
- `R_386_GOTPC` reloc :
  the distance from _here_(PC) to the GOT
  - used in the function prolog to calculate `&GOT[0]`

Dynamic Linking

- When dynamic linking is required
  (modern compiler set it by default),
  a compiler generates `.dynamic` section
- Note that executables and shared objects
  have a separate procedure linkage table (PLT)

http://dandylife.net/blog/archives/660

TOC: Introduction to a Relocatio Example

TOC: Relocation Example Library Code

Example library code

```
typedef struct {
  char* p;
  char (*f)(int);
} _st;

char fPub(int a) {
  return a;
}

static char fLocal(int b) {
  return b;
}

char cPub;             // uninitialized
static char cLocal;    // uninitialized
```

```
_st a[] = { { &cLocal,   // 1
              fLocal }, // 2
            { &cPub,     // 3
              fPub } }; // 4

int foo(int a) {         // 5
  return fPub(a)         // 6
       + fLocal(a)       // 7
       + (int) &cPub     // 8
       + cPub            // 9
       + (int) &cLocal   // 10
       + cLocal;         // 11
}
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Example executable code

- the main function code

```
extern int fPub(int);
extern int cPub;

int main() {
  return fPub(123)   // 1
       + cPub;       // 2
}
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Variable naming for this example

global	Pub
local	Local
function	f
character	c

	global	local
function	fPub	fLocal
character	cPub	cLocal

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Symbol defintions in this example

- global symbol defintions : fPub, cPub, a[], foo

```
fPub   : char fPub(int a) { return a; }
cPub   : char cPub;
a      : _st a[] = {{&cLocal, fLocal}, {&cPub, fPub}};
foo    : int foo(int a) { return fPub(a)+fLocal(a)+...+cLocal; }
```

- local symbol defintions : fLocal, cLocal

```
fLocal : static char fLocal(int b) { return b; }
cLocal : static char cLocal;
```

- function symbol definitions : fPub, fLocal, foo

```
fPub   : char fPub(int a) { return a; }
fLocal : static char fLocal(int b) { return b; }
foo    : int foo(int a) { return fPub(a)+fLocal(a)+...+cLocal; }
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Initialized vs unitialized global variables

- uninitialized global variables : .bss section

```
cPub   (global symbol)  : char cPub;
cLocal (local  symbol)  : static char cLocal;
```

- initialized global variable : .data section

```
a      (global symbol) : _st a[] = {{&cLocal, fLocal}, {&cPub, fPub}};
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Numbering symbol references in a[] and foo

1	`&cLocal`	static global variable	local	symbol ref
2	`fLocal`	static function address	local	symbol ref
3	`&cPub`	global variable address	global	symbol ref
4	`fPub`	function address	global	symbol ref
5	`foo`	function definition	global	symbol def
6	`fPub(a)`	function call	global	symbol ref
7	`fLocal(a)`	static function call	local	symbol ref
8	`&cPub`	global variable address	global	symbol ref
9	`cPub`	global variable	global	symbol ref
10	`&cLocal`	static global variable address	local	symbol ref
11	`cLocal`	static global variable	local	symbol ref

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html http://netwin

TOC: Symbol References

Symbol references in the library code `rel.c` (1)

```
_st a[] = { { &cLocal,  // 1   cLocal(1) in .data
              fLocal }, // 2   fLocal(1) in .data
            { &cPub,    // 3   cPub(1)   in .data
              fPub } }; // 4   fPub(1)   in .data

int foo(int a) {        // 5
  return fPub(a)        // 6   fPub(2)   in .text
       + fLocal(a)      // 7   fLocal(2) in .text
       + (int) &cPub    // 8   cPub(2)   in .text
       + cPub           // 9   cPub(3)   in .text
       + (int) &cLocal  // 10  cLocal(2) in .text
       + cLocal;        // 11  cLocal(3) in .text
}
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Symbol references in the library code `rel.c` (2)

```
_st a[]........&cLocal,    // 1   cLocal(1) in .data
foo......(int) &cLocal     // 10  cLocal(2) in .text
foo...........+ cLocal     // 11  cLocal(3) in .text

_st a[].......fLocal },    // 2   fLocal(1) in .data
foo.........+ fLocal(a)    // 7   fLocal(2) in .text


_st a[].......&cPub,       // 3   cPub(1)   in .data
foo.. ..(int) &cPub        // 8   cPub(2)   in .text
foo.. ......+ cPub         // 9   cPub(3)   in .text

_st a[].......fPub } };    // 4   fPub(1)   in .data
foo... return fPub(a)      // 6   fPub(2)   in .text
```

`http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html`

Symbol references in the executable code main.c (1)

```
extern int fPub(int);
extern int cPub;

int main() {
  return fPub(123)      // 1   fPub in .exec
       + cPub;          // 2   cPub in .exec
}
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Symbol references in the executable code `main.c` (2)

```
main.........+ cPub;      // 2   cPub in .exec

main.....return fPub(123)  // 1   fPub in .exec
```

```
http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html
```

Symbol references in .data section of the library

- symbol references in the .data section of the library

```
_st a[] = { { &cLocal,  // 1  cLocal local    sym reference
              fLocal }, // 2  fLocal function sym reference
            { &cPub,    // 3  cPub   global   sym reference
              fPub } }; // 4  fPub   function sym reference
```

- relocation information in the following sections
    - rel.data.rel (default)
    - rel.data.rel (-fPIC)
    - rel.data (-fno-pic)

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Symbol references in .text section of the library

- symbol references in the .text section of the librar

```
int foo(int a) {          // 5  foo    function sym defintion
  return fPub(a)          // 6  fPub   function sym reference
       + fLocal(a)        // 7  fLocal function sym reference
       + (int) &cPub      // 8  cPub   global   sym reference
       + cPub             // 9  cPub   global   sym reference
       + (int) &cLocal    // 10 cLocal local    sym reference
       + cLocal;          // 11 cLocal local    sym reference
}
```

- relocation information in .rel.text section

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Symbol references in .text section of the executable

- symbol references in the .text section of the executable

```
extern int fPub(int);
extern int cPub;

int main() {
  return fPub(123)      // 1   fPub function sym reference (call)
       + cPub;          // 2   cPub global   sym reference
}
```

- relocation information in .rel.text section

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Symbol value sections

fPub	function	global	symbol	in	`.text`
fLocal	static function	local	symbol	in	`.text`
cPub	global variable	global	symbol	in	`.bss`
cLocal	static global variable	local	symbol	in	`.bss`
a	global structure array	global	symbol	in	`.data`

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

.rel.text and .rel.data sections

- .rel.text : relocation information for .text section
    - a list of locations in the .text section
      that will need to be modified
      when the linker combines this object file with others
- .rel.data : relocation information for .data section
    - a list of locations in the .data section
      that will need to be modified
      when the linker combines this object file with others

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

`.rel.text` section

- relocation information for `.text` section
  - modify any instruction in the code section that
    - calls an external function
    - references a global variable
  - do not modify any instructions in the code section that
    - calls local functions
- executable files do not include relocation information unless the user explicitly instructs the linker

  `ld --emit-relocs, ld --relocatable`

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

.rel.data section

- relocation information for .data section
- relocation information for any global variable that are referenced or defined by the data section
- modify the initial values of any global variable
    - when the initial values are
        - the address of a global variable (e.g. &cPub)
        - externally defined function (e.g. &fPub)

```
_st a[] = { {&cLocal, fLocal}, {&cPub, fPub} }
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

.rel.text section output sample

- a list of locations in the .text section to be modified when the linker combines this object file with others

```
Relocation section '.rel.text' at offset 0x440 contains 6 entries:
Offset    Info   Type          Symbol's Value Symbol's Name
00000028 00e0a R_386_GOTPC   00000000       _GLOBAL_OFFSET_TABLE_
00000031 00a04 R_386_PLT32   00000000       fPub
0000003a 00604 R_386_PLT32   0000000c       fLocal
00000049 00c03 R_386_GOT32   00000001       cPub
00000057 00409 R_386_GOTOFF 00000000       .bss
0000005e 00409 R_386_GOTOFF 00000000       .bss
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

.rel.data section output sample

- a list of locations in the .data section to be modified when the linker combines this object file with others

```
Relocation section '.rel.data' at offset 0x470 contains 4 entries:
Offset    Info  Type      Symbol's Value Symbol's Name
00000000 00401 R_386_32 00000000        .bss
00000004 00201 R_386_32 00000000        .text
00000008 00c01 R_386_32 00000001        cPub
0000000c 00a01 R_386_32 00000000        fPub
```

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

TOC: Compiling scripts

- Using a shared library
- Script r1 (`rel.o` reloc analysis)
- Script r2 (`librel.so` reloc analysis)
- Script r3 (`main.o` reloc analysis)
- Script r4 (`run_dynamic` reloc analysis)

Using a shared library

- creating a shared library

```
gcc -m32 -fPIC -c -g rel.c
gcc -m32 -shared rel.o -o librel.so
```

- linking with a shared library

```
gcc -m32 -c -g main.c
gcc -m32 main.o -Wl,-q -L/home/young/ -lrel -o run_dynamic
```

- run a dynamic executable

```
LD_LIBRARY_PATH=/home/young/ ./run_dynamic
```

https://renenyffenegger.ch/notes/development/languages/C-C-plus-plus/GCC/create-libraries/index

Script r1 (rel.o reloc analysis)

```
gcc -m32 -c rel.c -o rel-default.o
readelf -s rel-default.o > r1-default.symtab
readelf -r rel-default.o > r1-default.reloc

gcc -m32 -fPIC -c rel.c -o rel-fPIC.o
readelf -s rel-fPIC.o > r1-fPIC.symtab
readelf -r rel-fPIC.o > r1-fPIC.reloc

gcc -m32 -fno-pic -c rel.c -o rel-fno-pic.o
readelf -s rel-fno-pic.o > r1-fno-pic.symtab
readelf -r rel-fno-pic.o > r1-fno-pic.reloc
```

Script r2 (`librel.so` reloc analysis)

```
gcc -m32 -c main.c -o main-default.o
objdump -t main-default.o > r2-default.symtab
readelf -r main-default.o > r2-default.reloc

gcc -m32 -fPIC -c main.c -o main-fPIC.o
objdump -t main-fPIC.o > r2-fPIC.symtab
readelf -r main-fPIC.o > r2-fPIC.reloc

gcc -m32 -fno-pic -c main.c -o main-fno-pic.o
objdump -t main-fno-pic.o > r2-fno-pic.symtab
readelf -r main-fno-pic.o > r2-fno-pic.reloc
```

Script r3 (main.o reloc analysis)

```
gcc -m32 -c rel.c -o rel-default.o
gcc -m32 -shared rel-default.o -o librel-default.so
objdump -tT librel-default.so > r3-default.symtab
readelf -r librel-default.so > r3-default.reloc

gcc -m32 -fPIC -c rel.c -o rel-fPIC.o
gcc -m32 -shared rel-fPIC.o -o librel-fPIC.so
objdump -tT librel-fPIC.so > r3-fPIC.symtab
readelf -r librel-fPIC.so > r3-fPIC.reloc

gcc -m32 -fno-pic -c rel.c -o rel-fno-pic.o
gcc -m32 -shared rel-fno-pic.o -o librel-fno-pic.so
objdump -tT librel-fno-pic.so > r3-fno-pic.symtab
readelf -r librel-fno-pic.so > r3-fno-pic.reloc
```

Script r4 (run_dynamic.so reloc analysis)

```
gcc -m32 -fPIC -c rel.c
gcc -m32 -shared rel.o -o librel.so

gcc -m32 -c main.c -o main-default.o
gcc -m32 main-default.o -Wl,-q -L/home/young/ -lrel -o run-default
LD_LIBRARY_PATH=/home/young/ ./run-default
objdump -T run-default > r4-default.symtab
readelf -r run-default > r4-default.reloc

gcc -m32 -fPIC -c main.c -o main-fPIC.o
gcc -m32 main-fPIC.o -Wl,-q -L/home/young/ -lrel -o run-fPIC
LD_LIBRARY_PATH=/home/young/ ./run-fPIC
objdump -T run-fPIC > r4-fPIC.symtab
readelf -r run-fPIC > r4-fPIC.reloc

gcc -m32 -fno-pic -c main.c -o main-fno-pic.o
gcc -m32 main-fno-pic.o -Wl,-q -L/home/young/ -lrel -o run-fno-pic
LD_LIBRARY_PATH=/home/young/ ./run-fno-pic
objdump -T run-fno-pic > r4-fno-pic.symtab
readelf -r run-fno-pic > r4-fno-pic.reloc
```