ELF1 6A Relocation - ELF document

Young W. Lim

2019-04-13 Sat

Outline

1. Based on

2. Relocation Entry and Relocation Table
    - Relocation Entries
    - Relocation Table

3. ELF Relocation Types
    - ELF relocation type summary
    - ELF relocation calculation
    - ELF relocation types
    - Multiple descriptions of R_386_GOT32

Based on

"Self-service Linux: Mastering the Art of Problem Determination",
Mark Wilding
"Computer Architecture: A Programmer's Perspective",
Bryant & O'Hallaron

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: Relocation Table

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

ELF Relocation Entry Members - `r_offset`

- the location at which to apply the relocation action
- symbol reference location

- for a relocatable file,
  - the offset value is the byte offset
    from the beginning of the section
    to the storage unit affected by the relocation

- for an executable or a shared object file,
  - the offset value is the virtual address of
    the storage unit affected by the relocation

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

ELF Relocation Entry Members - r_info

1. the symbol table index
   with respect to which the relocation must be made
   - application result of ELF32_R_SYM to r_info member
2. the relocation type to be applied
   - application result of ELF32_R_TYPE to r_info member

ELF32_R_TYPE, ELF32_R_SYM

```
#define ELF32_R_SYM(i)      ((i)>>8)
#define ELF32_R_TYPE(i)     ((unsigned char) (i))
#define ELF32_R_INFO(s,t)   (((s)<<8) + (unsigned char)(t))
```

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

ELF Relocation Entry Members - r_addend

- specifies a constant addend used to compute
    - the value to be stored into the relocation field
    - the symbol value to the symbol reference location
- only Elf32_Rela entries contain an explicit addend
- Elf32_Rel entries store an implicit addend in the location to be modified

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

ELF Relocation Section

- starts with a table of relocation entries which can be located using the relevant section header
- the section header
  - when sh_type is either SHT_REL or SHT_RELA
  - sh_link : the section header index of the associated symbol table
  - sh_info : the section header index of the section to which the relocation applies
- a relocation section references *two other sections*
  - a symbol table section
  - a section to modify a symbol reference

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

ELF Sections

- an object file's section header table lets one locate all the file's sections
- an array of Elf32_Shdr structures
- a section header table index is a subscript into this array
- ELF header members related to the section header table
  - e_shoff byte offset from the beginning of the file to the section header table
  - e_shnum : the number of entries the section header table contains
  - e_shentsize : the size in bytes of each entry

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

ELF Section Conditions (1)

- Every section in an object file has exactly one section header describing it
- Section headers may exist which do not have a section
- Each section occupies one contiguous (possibly empty) sequence of bytes within a file

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

ELF Section Conditions (2)

- Sections in a file may not overlap
  No byte in a file resides in more than one section
- An object file may have inactive space
  The various headers and the sections might not cover every byte in an object file
  The contents of the inactive data are unspecified

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

ELF Section Header Structure

ELF Section Header Structure : Elf32_Shdr

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

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

ELF Relocation Entries for relocatable object files

relocatable files

- `r_offset` holds a section offset
- the relocation section itself describes
  how to modify another section in the file
  - relocation entries in a relocation table
- the relocation offsets designate a storage unit
  within the second section (symbol reference)

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

ELF Relocation Entries for executable and shared object files

executable and shared object files

- r_offset holds a virtual address
- to make these files' relocation entries more useful for dynmic linker
- the section offset (file interpretation) gives way to a virtul address (memory interpretation)

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

Columns of readelf -r (1)

readelf -r /bin/ls | head -n 20

```
Relocation section '.rela.dyn' at offset 0x15b8 contains 7 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000619ff0  003e00000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0

Relocation section '.rela.plt' at offset 0x1660 contains 105 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
00000061a018  000100000007 R_X86_64_JUMP_SLO 0000000000000000 __ctype_toupper_loc +
```

https://stackoverflow.com/questions/19593883/understanding-the-relocation-table-ou

Columns of readelf -r (2)

- `readelf -r /bin/ls | head -n 20`

Offset	000000619ff0
Info	003e00000006
Type	R_X86_64_GLOB_DAT
Sym. Value	0000000000000000
Sym. Name + Addend	__gmon_start__ + 0

https://stackoverflow.com/questions/19593883/understanding-the-relocation-table-ou

Relocation Table

Offset	where the symbol value should go
Info	- the type (the exact calculation depends on the arch) - the symbol index in the symtab
Type	relocation type of the symbol according to the ABI
Sym value	the addend to be added to the symbol resolution
Sym name Addend	a pretty printing of the symbol name + addend.

```
Relocation section '.rela.dyn' at offset 0x15b8 contains 7 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000619ff0  003e00000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0
```

https://stackoverflow.com/questions/19593883/understanding-the-relocation-table-ou

Relocation Table Example

readelf -r swap.o

```
Relocation section ’.rel.text’ at offset 0x478 contains 8 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
00000007  00001402 R_386_PC32        00000000   __x86.get_pc_thunk.ax
0000000c  0000150a R_386_GOTPC       00000000   _GLOBAL_OFFSET_TABLE_
00000012  0000122b R_386_GOT32X      00000004   p1
00000018  0000112b R_386_GOT32X      00000000   buf
00000023  00001009 R_386_GOTOFF      00000000   p0
0000002e  0000122b R_386_GOT32X      00000004   p1
00000036  00001009 R_386_GOTOFF      00000000   p0
00000040  0000122b R_386_GOT32X      00000004   p1
```

Relocation Example other Reference

- https://wiki.osdev.org/ELF_Tutorial

(1) ELF relocation types

name	value	field	calculation
R_386_NONE	0	None	None
R_386_32	1	word32	S+A
R_386_PC32	2	word32	S+A-P
R_386_GOT32	3	word32	G+A-GOT
R_386_PLT32	4	word32	L+A-P
R_386_COPY	5	None	None
R_386_GLOB_DAT	6	word32	S
R_386_JMP_SLOT	7	word32	S
R_386_RELATIVE	8	word32	B+A
R_386_GOTOFF	9	word32	S+A-GOT
R_386_GOTPC	10	word32	GOT+A-P
R_386_32PLT	11	word32	L+A

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(2) A, B

- A represents the addend used to compute the value of the relocatable field.
- B represents the base address at which a shared object has been loaded into memory during execution.
  - generally, a shared object is built with a 0 base virtual address, but the execution address will be different.
  - R_386_RELATIVE $(B + A)$

https://refspecs.linuxfoundation.org/elf/x86_64-abi-0.95.pdf

(3) G, GOT

- G represents the offset into the global offset table (GOT) entry at which the relocation entry's symbol will reside during execution.
    - address of a GOT entry
    - `R_386_GOT32` $(G + A - GOT)$
- GOT represents the address of the global offset table(GOT).
    - base address of GOT
    - `R_386_GOTOFF` $(S + A - GOT)$
    - `R_386_GOTPC` $(GOT + A - P)$

https://refspecs.linuxfoundation.org/elf/x86_64-abi-0.95.pdf

(4) L

- L represents the place (section offset or address) of the Procedure Linkage Table (PLT) entry for a symbol.
  - address of a PLT entry
  - R_386_PLT32 $(L + A - P)$
  - R_386_32PLT $(L + A)$

https://refspecs.linuxfoundation.org/elf/x86_64-abi-0.95.pdf

(5) P, S

- P represents the place (section offset or address) of the storage unit being relocated (computed using `r_offset`).
  - address of a symbol reference
  - `R_386_PC32` $(S + A - P)$
  - `R_386_PLT32` $(L + A - P)$

- S represents the value of the symbol whose index resides in the relocation entry
  - address of a symbol
  - `R_386_32` $(S + A)$
  - `R_386_PC32` $(S + A - P)$
  - `R_386_GLOB_DAT` (S)
  - `R_386_JMP_SLOT` (S)

`https://refspecs.linuxfoundation.org/elf/x86_64-abi-0.95.pdf`

(6) PIC relocs

GOT32	$(G + A - GOT)$	GOT-relative, GOT entry address
PLT32	$(L + A - P)$	PC-relative, PLT entry address
32PLT	$(L + A)$	PLT entry address
GOTOFF	$(S + A - GOT)$	GOT-relative, symbol address symbol's GOT OFFSET
GOTPC	$(GOT + A - P)$	PC-relative, GOT address GOT address w.r.t. PC

G, L, GOT

- G : entry address of the GOT
- L : entry address of the PLT
- GOT : base address of the GOT

https://refspecs.linuxfoundation.org/elf/x86_64-abi-0.95.pdf

(7) Offset relocs

R_386_	calc	
COPY	None	r_offset: WR segment location
GLOB_DAT	S	r_offset: GOT entry location
JMP_SLOT	S	r_offset: PLT entry location
RELATIVE	B+A	r_offset: offset in shared object

- r_offset :
  the location which the linker will fill in
  normally the symbol reference location

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(8) My ELF relocation type table

R_386_	calc	
32	S+A	absolute
PC32	S+A-P	pc-relative symbol address
PLT32	L+A-P	pc-relative plt entry address
32PLT	L+A	plt entry address
GOT32	G+A-GOT	got-relative got entry address
GOTOFF	S+A-GOT	got-relative symbol address
GOTPC	GOT+A-P	pc-relative got address
NONE	None	
COPY	None	r_offset: WR segment location
GLOB_DAT	S	r_offset: GOT entry location
JMP_SLOT	S	r_offset: PLT entry location
RELATIVE	B+A	r_offset: offset in shared object

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(Type 1) R_386_32 Formula

R_386_32

- $(S + A)$
- absolute symbol address
  - S represents the value of the symbol whose index resides in the relocation entry
  - A represents the addend used to compute the value of the relocatable field.

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(Type 2) R_386_PC32 Formula

R_386_PC32

- $(S + A - P)$
- PC-relative, symbol address
  - S represents the value of the symbol whose index resides in the relocation entry
  - A represents the addend used to compute the value of the relocatable field.
  - P represents the place (section offset or address) of the storage unit being relocated (computed using `r_offset`).

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(Type 3) R_386_GOT32 Formula

R_386_GOT32

- $(G + A - GOT)$
- GOT-relative, GOT entry address
  - G represents the offset into the global offset table(GOT) at which the relocation entry's symbol will reside during execution.
  - A represents the addend used to compute the value of the relocatable field.
  - GOT represents the address of the global offset table(GOT).

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(Type 4) R_386_PLT32 Formula

R_386_PLT32

- $(L + A - P)$
- PC-relative, PLT entry address
  - L represents the place (section offset or address) of the Procedure Linkage Table (PLT) entry for a symbol.
  - A represents the addend used to compute the value of the relocatable field.
  - P represents the place (section offset or address) of the storage unit being relocated (computed using `r_offset`).

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(Type 8) R_386_RELATIVE Formula

R_386_RELATIVE

- $(B + A)$
- base and offset addresses
  - B represents the base address
    at which a shared object has been loaded
    into memory during execution.
    - generally, a shared object is built
      with a 0 base virtual address,
      but the execution address will be different.
  - A represents the addend used to compute w
    the value of the relocatable field.

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(Type 9) R_386_GOTOFF Formula

R_386_GOTOFF

- $(S + A - GOT)$
- GOT-relative, symbol address
  - S represents the value of the symbol whose index resides in the relocation entry
  - A represents the addend used to compute the value of the relocatable field.
  - GOT represents the address of the global offset table(GOT).

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(Type 10) R_386_GOTPC Formula

R_386_GOTPC

- $(GOT + A - P)$
- PC-relative, GOT address
  - GOT represents the address of the global offset table(GOT).
  - A represents the addend used to compute the value of the relocatable field.
  - P represents the place (section offset or address) of the storage unit being relocated (computed using `r_offset`).

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(Type 11) R_386_32PLT Formula

R_386_32PLT

- $(L + A)$
- absolute PLT entry address
  - L represents the place (section offset or address) of the Procedure Linkage Table (PLT) entry for a symbol.
  - A represents the addend used to compute the value of the relocatable field.

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(1) R_386_GOT32 ($G + A - GOT$)

- computes the distance from the base of the GOT
  to the symbol's GOT entry
- GOT-relative, GOT entry address
- it also instructs the link editor to create a global offset table.
- the relative location of the slot (entry) in the GOT
  where the linker has placed a pointer to the given symbol
  it is used for indirectly referenced global data
  (- Linkers and Loaders, J. R. Levine)

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(2) R_386_PLT32 ($L + A - P$)

- computes the address of the symbol's PLT entry
- PC-relative, PLT entry address

- instructs the link editor to create a procedure linkage table.

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(3) R_386_COPY (None)

- created by the link editor for dynamic executables
  to preserve a read-only text segment.
- its offset member refers to a location in a writable segment.
- during execution, the runtime linker copies
  data associated with the shared object's symbol
  to the location specified by the offfset
- The symbol table _index specifies a symbol that should exist
  both in the current object file and in a shared object.

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(4) R_386_GLOB_DAT (S)

- used to set a GOT entry to the address of the specified symbol
- the special relocation type enable you to determine the correspondence between symbols and GOT entries
- its offset member gives the location of a GOT entry.

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(5) R_386_JMP_SLOT (S)

- created by the link editor for dynamic objects
  to provide lazy binding
- used to set a PLT entry to the address of the symbol
  through a GOT entry
- the runtime linker modifies the GOT entry
  to transfer control to the designated symbol address
- its offset member gives the location
  of a PLT entry.

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(6) R_386_RELATIVE ($B + A$)

- created by the link editor for shared dynamic objects.
- its offset member gives the location within a shared object that contains a value representing a relative address. (r_offset, r_info, r_addend)
- the runtime linker computes the corresponding virtual address by adding the virtual address at which the shared object is loaded to the relative address.
- virtual address (base) + relative address (offset)
- relocation entries for this type must specify 0 for the symbol table index.
- this is used to mark data addresses in a PIC shared library that need to be relocated at load time (- Linkers and Loaders, J. R. Levine)

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(7) R_386_GOTOFF ($S + A - GOT$)

- Computes the difference between a symbol's value and the address of the GOT
- GOT-relative, symbol address
- It also instructs the link editor to create the global offset table.
- the distance form the base of the GOT to the given symbol or address it is used to address static data (local symbols)

(Linkers and Loaders, J. R. Levine)

`https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html`

(8) R_386_GOTPC ($GOT + A - P$)

- Resembles R_386_PC32, except that it uses the address of the GOT in its calculation.
- PC-relative, GOT address
- The symbol referenced in this relocation normally is GLOBAL_OFFSET_TABLE,
- also instructs the link-editor to create the GOT.

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

R_386_GOT32 : multiple descriptions (1)

- multiple descriptions
    1. "System V Application Binary Interface Intel386 Architecture Processor Supplement Version 1.1" (https://github.com/hjl-tools/x86-psABI/wiki/intel386-psABI-1.1.pdf), p36 contains next calculation for R_386_GOT32: G + A - GOT.
    2. SYSTEM V APPLICATION BINARY INTERFACE 4 (https://refspecs.linuxfoundation.org/elf/abi386-4.pdf, p78) tolds us its G + A - P.
    3. Oracle docs (https://docs.oracle.com/cd/E19455-01/816-0559/chapter6-26/index.html) says its should be G + A.
    4. gold/bfd calculates it as (gotentryaddr - gotsize + A), so it is some negative offset.

Patch implements gold/bfs behavior to be consistent with.

https://reviews.llvm.org/D15750?id=43537

R_386_GOT32 : multiple descriptions (2)

intel386-psABI-1.1.pdf

- R_386_GOT32 $(G + A - GOT)$
- R_386_PLT32 $(L + A - P)$
- R_386_GOT32X $(G + A - GOT \ / \ G + A - GOT)$
  used without base register when PIC is disabled

https://reviews.llvm.org/D15750?id=43537

R_386_GOT32 : multiple descriptions (3)

- $G + A - GOT$ is seemed more accurate than $G + A - GOT$
- $G + A - P$ is preferable to me, regarding $L + A - P$
- either $G + A - GOT$ or $G + A - P$ will be used

intel386-psABI-1.1.pdf

- `R_386_GOT32` $(G + A - GOT)$
- `R_386_PLT32` $(L + A - P)$
- `R_386_GOT32X` $(G + A - GOT \ / \ G + A - GOT)$
  used without base register when PIC is disabled

`https://reviews.llvm.org/D15750?id=43537`