ELF1 7 Examples - 3 Executables - ELF Study 1999

Young W. Lim

2019-11-13 Wed

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
- -Wl,-q

TOC: PIC relocs background

- Relocs background in shared object and executable files
- Relocs in an object file
- Relocs in a shared object file
- Relocs in an executable file

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 symbol reference 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 a global symbol reference in PIC shared libraries
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 a function symbol reference in PIC shared libraries

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

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 global symbol • offset : a PLT entry location of a *PIC* shared library • fill the location 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)
  - GOT is utilized, but PLT is not
    for function definitions in the same module
  - GOT and PLT are utilized
    for function references, that are defined externally
- to use both GOT and PLT, use `-fPIC`
- to disable the PIC feature use `-fno_pic`
  then neither GOT nor PLT is used
  - neither `R_386_PLT32` or `R_386_GOT32` reloc is used
  - only `R_386_32` and `R_386_PC32` are used

Dynamic Lin

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

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

TOC: Executable reloc exammple using a shared library

- Example codes
- Compling scripts

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

Symbol references in the library code (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 (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 (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 (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

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

Scrit r1 (producing rel.o and analyzing relocations)

```
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
```

Scrit r2 (producing librel.so and analyzing relocations)

```
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
```

Scrit r3 (producing main.o and analyzing relocations)

```
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
```

Scrit r4 (producing run_dynamic.so and analyzing relocations)

```
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
```

TOC: Relocation Results Summary

1. `rel.o` object file relocs
2. `librel.so` shared object file relocs
3. `main.o` object file relocs
4. `run_dynamic` executable file relocs

1.a data section relocs of an object file rel.o

- global data symbol reference (cPub) in .data
  - R_386_32 (-fno-pic, default, -fPIC)
- global function symbol reference (fPub) in .data
  - R_386_32 (-fno-pic, default, -fPIC)

1.b .rel.data.rel relocs of rel.o

- .rel.data relocs of rel.o file (-fno-pic)
  .rel.data.rel relocs of rel.o file (default)
  .rel.data.rel relocs of rel.o file (-fPIC)

	-fno-pic	default	-fPIC
.bss	R_386_32 absolute	R_386_32 absolute	R_386_32 absolute
.text	R_386_32 absolute	R_386_32 absolute	R_386_32 absolute
.cPub	R_386_32 absolute	R_386_32 relative to .bss	R_386_32 relative to .bss
.fPub	R_386_32 absolute	R_386_32 relative to .text	R_386_32 relative to .text

1.c text section relocs of an object file rel.o

- global data symbol reference (cPub) in .text
  - R_386_GOT32 : when GOT is used (default, -fPIC)
  - R_386_32 : otherwise (-fno-pic)
- global function symbol reference (fPub) in .text
  - R_386_PLT32 : when PLT is used (-fPIC)
  - R_386_PC32 : otherwise (-fno-pic, default)

1.d .rel.text relocs of rel.o

	-fno-pic	default	-fPIC
cPub	R_386_32 absolute	R_386_GOT32x GOT	R_386_GOT32x GOT
fPub	R_386_PC32 relative	R_386_PC32 relative	R_386_PLT32 PLT

2.a data section relocs of a shared object librel.so

- global data symbol reference (cPub) in .data
  - R_386_32 : always (-fno-pic, default, -fPIC)
- global function symbol reference (fPub) in .data
  - R_386_32 : always (-fno-pic, default, -fPIC)

2.b text section relocs of a shared object librel.so

- global data symbol reference (cPub) in .text
  - R_386_GOT32 → R_386_GLOB_DAT :
    when GOT is used (-fPIC, default)
  - R_386_32 : otherwise (-fno-pic)
- global function symbol reference (fPub) in .text
  - R_386_PLT32 → R_386_JUMP_SLOT :
    when PLT is used (-fPIC)
  - R_386_PC32 : otherwise (-fno-pic, default)

2.c .rel.dyn relocs of librel.so

	-fno-pic	default	-fPIC
cPub	R_386_32	R_386_GLOB_DAT	R_386_GLOB_DAT
	R_386_32	R_386_32	R_386_32
	R_386_32		
	absolute	GOT, absolute	GOT, absolute
fPub	R_386_PC32	R_386_PC32	R_386_32
	R_386_32	R_386_32	
	relative, absolute	relative, absolute	absolute

2.d .rel.plt relocs of librel.so

	-fno-pic	default	-fPIC
fPub			R_386_JUMP_SLOT
	not applicable	not applicable	PLT

3.a text section relocs of an object file main.o

- global data symbol (cPub) reference in .text
  - R_386_GOT32 : when GOT is used (default, -fPIC)
  - R_386_32 : otherwise (-fno-pic)
- global function symbol (fPub) reference in .text
  - R_386_PLT32 : when PLT is used (default, -fPIC)
  - R_386_PC32 : otherwise (-fno-pic)

3.b .rel.text relocs of main.o file

	-fno-pic	default	-fPIC
cPub	R_386_32	R_386_GOT32x	R_386_GOT32x
	absolute	GOT	GOT
fPub	R_386_PC32	R_386_PLT32	R_386_PLT32
	relative	PLT	PLT

4.a Dynamic relocation section

- The dynamic relocation section describes
  all locations within the object that must be adjusted
  if the object is loaded at an address
  other than its linked base address.
- Only one dynamic relocation section is used
  to resolve addresses in data items,
  and it must be called `.rel.dyn`

https://www3.physnet.uni-hamburg.de/physnet/Tru64-Unix/HTML/APS31DTE/DOCU_002.HTM

4.b .rel.text and .rel.dyn

- shared (dynamic) executable files can contain normal relocation sections (`.rel.text`) in addition to a dynamic relocation section (`.rel.dyn`, `.rel.plt`)
- the normal relocation sections (`.rel.text`) may contain resolutions for any absolute values in the main program.
- the dynamic linker does not resolve these or relocate the main program.

https://www3.physnet.uni-hamburg.de/physnet/Tru64-Unix/HTML/APS31DTE/DOCU_002.HTM#

4.c normal relocs of a dynamic executable run_dynamic

- normal relocation sections (.rel.text)
- global data symbol reference (cPub) in .text
  - R_386_GOT32 : when GOT is used (-fno-pic, default, -fPIC)
- global function symbol reference (fPub) in .text
  - R_386_PLT32 : when PLT is used (default, -fPIC)
  - R_386_PC32 : otherwise (-fno-pic)

4.d dynamic relocs of a dynamic executable run_dynamic

- dynamic relocation section (.rel.dyn, .rel.plt)
- global data symbol reference (cPub) in .text
  - R_386_GOT32 → R_386_GLOB_DAT:
    when GOT is used (default, -fPIC)
  - R_386_32, R_386_COPY : otherwise (-fno-pic)
- global function symbol reference (fPub) in .text
  - R_386_PLT32 → R_386_JUMP_SLOT :
    when PLT is used (default, -fPIC)
  - R_386_PC32 : otherwise (-fno-pic)

4.e .rel.text relocs of run_dynamic

	-fno-pic	default	-fPIC
cPub	R_386_GOT32x GOT	R_386_GOT32x GOT	R_386_GOT32x GOT
fPub	R_386_PC32 relative	R_386_PLT32 PLT	R_386_PLT32 PLT

4.f .rel.plt relocs of run_dynamic

	-fno-pic	default	-fPIC
fPub	R_386_JUMP_SLOT PLT	R_386_JUMP_SLOT PLT	R_386_JUMP_SLOT PLT

4.g .rel.dyn relocs of run_dynamic

	-fno-pic	default	-fPIC
cPub	R_386_32 R_386_COPY	R_386_GLOB_DAT	R_386_GLOB_DAT
	absolute	GOT	GOT
fPub	R_386_PC32		
	relative		

4.e calling fPub in main.o

- main.o with -fno-pic

```
16:   e8 fc ff ff ff          call   17 <main+0x17>
                      17: R_386_PC32  fPub
```

- main.o with default

```
1f:   e8 fc ff ff ff          call   20 <main+0x20>
                      20: R_386_PLT32 fPub
```

- main.o with -fPIC

```
1f:   e8 fc ff ff ff          call   20 <main+0x20>
                      20: R_386_PLT32 fPub
```

4.f calling fPub in run-dynamic

- run-dynamic with -fno-pic

```
613:  e8 fc ff ff ff          call   614 <main+0x17>
                      614: R_386_PC32 fPub
```

- run_dynamic with default

```
5fc:  e8 7f fe ff ff          call   480 <fPub@plt>
                      5fd: R_386_PLT32        fPub
```

- run_dynamic with -fPIC

```
5fc:  e8 7f fe ff ff          call   480 <fPub@plt>
                      5fd: R_386_PLT32        fPub
```

gcc -Wl,-q is used

4.g referencing cPub in main.o

- main.o with -fno-pic

```
20:   a1 00 00 00 00          mov    0x0,%eax
                      21: R_386_32    cPub
```

- main.o with default

```
29:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      2b: R_386_GOT32X        cPub
```

- main.o with -fPIC

```
29:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      2b: R_386_GOT32X        cPub
```

4.h referencing cPub in run-dynamic

- run-dynamic with -fno-pic

```
61d:  a1 00 00 00 00          mov    0x0,%eax
                      61e: R_386_32   cPub
```

- run_dynamic with default

```
606:  8b 83 24 00 00 00       mov    0x24(%ebx),%eax
                      608: R_386_GOT32X       cPub
```

- run_dynamic with -fPIC

```
606:  8b 83 24 00 00 00       mov    0x24(%ebx),%eax
                      608: R_386_GOT32X       cPub
```

gcc -Wl,-q is used

TOC: Relocations in shared object (.so) files

- Linking the `.data` section
- Linking the `.text` section
- Relocation Summary in `rel.o`
- Relocation Summary in `librel.so`

TOC: linking the .data section

- resolving local symbol references &cLocal, fLocal
- resolving global symbol references &cPUb, fPub

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

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

.data.rel (1)

- gcc, the GNU linker, and the glibc dynamic linker cooperate to implement read-only relocations, or relro.
- This permits the linker to designate a part of an executable or (more commonly) a shared library as being read-only after dynamic relocations have been applied.

https://stackoverflow.com/questions/7029734/what-is-the-data-rel-ro-used-for

.data.rel (2)

- This may be used for read-only global variables
  which are initialized to something which requires a relocation,
  such as the address of a function or a different global variable.
- Because the global variable requires a runtime initialization
  in the form of a dynamic relocation, it can not be placed
  in a read-only segment.
- However, because it is declared to be constant,
  and therefore may not be changed by the program,
  the dynamic linker can mark it as read-only
  after the dynamic relocation has been applied.

https://stackoverflow.com/questions/7029734/what-is-the-data-rel-ro-used-for

resolving local symbol references (1)

- when the linker is called for the final link stage
  - `cLocal` is in the `.bss` section (uninitialized)
  - `fLocal` is in the `.text` section (function)
  - `.bss` has the `R_386_32` reloc
  - `.text` has the `R_386_32` reloc
  - the reloc targets are `&cLocal`, `fLocal`
- `R_386_32` reloc will be changed into a `R_386_RELATIVE` for `.bss` and `.text`
- the offset is stored at the reloc target location

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

resolving local symbol references (2)

- At the beginning of the run time,
  - `.bss` has now the `R_386_RELATIVE` reloc
  - `.text` has now the `R_386_RELATIVE` reloc
  - the reloc targets are `&cLocal`, `fLocal`
  - the offset is stored at the reloc target location
- the dynamic linker will
  - add the module (base) address to the offset and
  - store the added result at the reloc target
- the *symbol name* is not used in this case
- but the *section number* is used instead

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

resolving local symbol references (3)

- relocs for `&cLocal` and `fLocal` depend on the compile options

	section	default	-fPIC
`&cLocal`	`.bss`	`R_386_32`	`R_386_RELATIVE`
`fLocal`	`.text`	`R_386_32`	`R_386_RELATIVE`

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

a calling fPub in rel.o

- rel.o with -fno-pic

```
17:   e8 fc ff ff ff          call   18 <foo+0x8>
                      18: R_386_PC32  fPub
```

- rel.o with default

```
37:   e8 fc ff ff ff          call   38 <foo+0x14>
                      38: R_386_PC32  fPub
```

- rel.o with -fPIC

```
3a:   e8 fc ff ff ff          call   3b <foo+0x17>
                      3b: R_386_PLT32 fPub
```

b calling fPub in librel.so

- librel.so with -fno-pic

```
000004ad <fPub>:
000004ad <fPub>:
4c4:   e8 fc ff ff ff          call   4c5 <foo+0x8>
```

- librel.so with default

```
0000049d <fPub>:
000004c1 <foo>:
4d4:   e8 fc ff ff ff          call   4d5 <foo+0x14>
```

- librel.so with -fPIC

```
000004ad <fPub>:
000004d1 <foo>:
4e7:   e8 a4 fe ff ff          call   390 <fPub@plt>
```

c referencing cPub in rel.o

- rel.o with -fno-pic

```
32:   ba 00 00 00 00          mov    $0x0,%edx
                      33: R_386_32    cPub

39:   0f b6 05 00 00 00 00    movzbl 0x0,%eax
                      3c: R_386_32    cPub
```

- rel.o with default

```
53:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      55: R_386_GOT32X        cPub

5b:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      5d: R_386_GOT32X        cPub
```

- rel.o with -fPIC

```
59:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      5b: R_386_GOT32X        cPub

61:   8b 83 00 00 00 00       mov    0x0(%ebx),%eax
                      63: R_386_GOT32X        cPub
```

d referencing cPub in librel.so

- librel.so with -fno-pic

```
4df:  ba 00 00 00 00          mov    $0x0,%edx
4e4:  01 c2                   add    %eax,%edx
4e6:  0f b6 05 00 00 00 00    movzbl 0x0,%eax
```

- librel.so with default

```
4f0:  8b 83 f4 ff ff ff       mov    -0xc(%ebx),%eax
4f6:  01 c2                   add    %eax,%edx
4f8:  8b 83 f4 ff ff ff       mov    -0xc(%ebx),%eax
```

- librel.so with -fPIC

```
506:  8b 83 f4 ff ff ff       mov    -0xc(%ebx),%eax
50c:  01 c2                   add    %eax,%edx
50e:  8b 83 f4 ff ff ff       mov    -0xc(%ebx),%eax
```

e Section header qqq in librel.so

```
readelf -S librel-fPIC.so

Section Headers:
[Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
[ 3] .dynsym           DYNSYM          00000180 000180 0000e0 10   A  4   1  4
[ 5] .rel.dyn          REL             000002e8 0002e8 000060 08   A  3   0  4
[ 6] .rel.plt          REL             00000348 000348 000008 08  AI  3  18  4
[ 8] .plt              PROGBITS        00000380 000380 000020 04  AX  0   0 16
[ 9] .plt.got          PROGBITS        000003a0 0003a0 000010 08  AX  0   0  8
[10] .text             PROGBITS        000003b0 0003b0 00018b 00  AX  0   0 16
[16] .dynamic          DYNAMIC         00001f2c 000f2c 0000c0 08  WA  4   0  4
[17] .got              PROGBITS        00001fec 000fec 000014 04  WA  0   0  4
[18] .got.plt          PROGBITS        00002000 001000 000010 04  WA  0   0  4
[19] .data             PROGBITS        00002010 001010 000014 00  WA  0   0  4
[20] .bss              NOBITS          00002024 001024 000004 00  WA  0   0  1
[22] .symtab           SYMTAB          00000000 001050 000390 10     23  44  4
```

f Relocs in librel.so

```
  readelf -r librel-fPIC.so

Relocation section '.rel.dyn' at offset 0x2e8 contains 12 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
00001f24  00000008 R_386_RELATIVE
00001f28  00000008 R_386_RELATIVE
00002010  00000008 R_386_RELATIVE
00002014  00000008 R_386_RELATIVE                cLocal
00002018  00000008 R_386_RELATIVE                fLocal
00001ff4  00000a06 R_386_GLOB_DAT    00002026    cPub
0000201c  00000a01 R_386_32          00002026    cPub
00002020  00000901 R_386_32          000004ad    fPub

Relocation section '.rel.plt' at offset 0x348 contains 1 entry:
 Offset     Info    Type            Sym.Value  Sym. Name
0000200c  00000907 R_386_JUMP_SLOT   000004ad    fPub
```

g Offset in librel.so

```
00001f24  1f2c .dynamic -0008 00000008 R_386_RELATIVE
00001f28  1f2c .dynamic -0004 00000008 R_386_RELATIVE
00001ff4  1f2c .dynamic  00c8 00000a06 R_386_GLOB_DAT    00002026   cPub
00002010  2010 .data     0000 00000008 R_386_RELATIVE
00002014  2010 .data     0004 00000008 R_386_RELATIVE               cLocal
00002018  2010 .data     0008 00000008 R_386_RELATIVE               fLocal
0000201c  2010 .data     000c 00000a01 R_386_32          00002026   cPub
00002020  2010 .data     0010 00000901 R_386_32          000004ad   fPub

Relocation section '.rel.plt' at offset 0x348 contains 1 entry:
 Offset     Info    Type            Sym.Value  Sym. Name
0000200c  00000907 R_386_JUMP_SLOT   000004ad   fPub

000004bf <fLocal>:
 4f8:   e8 c2 ff ff ff          call   4bf <fLocal>
```

h .data.rel in rel.o

```
00000000 <a>:
   0:   00 00                   add    %al,(%eax)
                        0: R_386_32     .bss        cLocal
   2:   00 00                   add    %al,(%eax)
   4:   12 00                   adc    (%eax),%al
                        4: R_386_32     .text       fLocal
        ...
                        8: R_386_32     cPub
                        c: R_386_32     fPub
```

i .data in librel.so

```
Desensamblado de la sección .data:

00002010 <__dso_handle>:
    2010:       10 20                       adc    %ah,(%eax)
        ...

00002014 <a>:
    2014:       25 20 00 00 bf              and    $0xbf000020,%eax
    2019:       04 00                       add    $0x0,%al
        ...
```

j .data.rel in rel.o

```
objdump -s -j .data.rel rel-fPIC.o

rel-fPIC.o:     file format elf32-i386

Contents of section .data.rel:
 0000 00000000 12000000 00000000 00000000  ................

readelf -x .data.rel rel-fPIC.o

Hex dump of section '.data.rel':
 NOTE: This section has relocations against it,
 but these have NOT been applied to this dump.
  0x00000000 00000000 12000000 00000000 00000000 ................

  https://stackoverflow.com/questions/1685483/how-can-i-examine-contents-of-a-data-s
```

k .data.rel in librel.so

```
objdump -s -j .data librel-fPIC.so

librel-fPIC.so:     file format elf32-i386

Contents of section .data:
 2010 10200000 25200000 bf040000 00000000  . ..% ..........
 2020 00000000                             ....


readelf -x .data librel-fPIC.so

Hex dump of section '.data':
  0x00002010 10200000 25200000 bf040000 00000000 . ..% ..........
  0x00002020 00000000                            ....

  https://stackoverflow.com/questions/1685483/how-can-i-examine-contents-of-a-data-s
```

TOC: linking the .text section

- resolving function symbol definitions `foo`
- resolving function symbol references `fPub(a)`, `fLocal(a)`
- resolving global symbol references `&cPub`, `cPub`
- resolving local symbol references `&cLocal`, `cLocal`

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

resolving public function symbol definition

- `foo(int a)` reloc is fixed up fully,
  does no appear in the library
- the function `foo` as a public symbol
  which is called externally (outside of the library)
- PIC / PIE requires a local reference to the GOT
  - `R_386_GOTPC` reloc for `&GOT[0]`
    the distance from here to the GOT

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

resolving global function symbol references

- the reloc of a global function reference will cause the linker to add a PLT entry and a corresponding GOT entry
  - the reloc of `fPub(a)` is translated into a indirect call through the PLT entry
  - the GOT entry gets a `R_386_JUMP_SLOT` reloc using the symbol `fPub`

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

resolving local function symbol references

- the reloc of a local function reference is converted into a direct call to the function
  - the reloc of `fLocal(a)` is converted into a direct call to `fLocal()`
  - because it can be fully resolved at the final linker stage

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

resolving global data symbol references

- the relocs of a global data symbol reference will cause the linker to add a GOT entry to hold them
- the relocs at &cPub (address) and cPub (data) will have an GOT entry to hold &cPub
  - the symbol value is an address of the symbol
- the GOT entry is marked with a R_386_GLOB_DAT reloc asking the dynamic linker for the full 32-bit absolute address

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

resolving local data symbol references

- the relocs of local data symbol references are fully resolved at final link time
- the relocs at `&cLocal` (address) and `cLocal` (data) are not required
- `&cLocal` can be computed as `&GOT[ 0]` plus the offset
- find the data at `&GOT[ 0]` plus the offset

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

resolving global symbol references (non-PIC) (1)

- for a non-PIC
  - &cPub will retain R_386_32 relocs and
  - fPub will retain R_386_PC32 relocs and
  - the dynamic linker will store at the reloc target the full 32-bit absolute and relative addresses

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

resolving global symbol references (PIE) (2)

- for a PIE (default)
  - &cPub will retain R_386_GOT32 relocs and
  - fPub will retain R_386_PC32 relocs and
  - the GOT is used for global symbols
  - the PLT is not used for function symbols

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

resolving global symbol references (PIC) (3)

- for a PIC
  - &cPub will retain R_386_GOT32 relocs and
  - fPub will retain R_386_PLT32 relocs and
  - the GOT is used for global symbols
  - the PLT is used for function symbols

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

Summary

- the 10 relocs turn into 4 relocs in the library
- the PLT gets a new entry
- the GOT gets two new entries

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

TOC: Relocations in an executable file

- Executable code example
- Relocations in an executable file
- Relocation Summary in `main.o`
- Relocation Summary in `run_dynamic`

TOC: Relocations in an executable file

- relocs for `fPub(123)` and cPub
- relocs for cPub

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

relocs for fPub(123) - global function symbol referefence

- when the executable is created,
  the R_386_PC32 at fPub(123)
  will have an PLT entry location of the shared library
- call to the PLT entry will be performed first
- the GOT entry in the shared library
  will get a R_386_JUMP_SLOT reloc
  using fPub symbol

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

relocs for cPub : global data symbol reference (1)

- the data reference of cPub will cause a local copy
  of the global cPub to be created in the data space of the app
- the data reference of cPub is changed
  to point to this new global data, and
  the reloc is resolved
- this new global gets a R_386_COPY reloc,
  using the symbol cPub

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

relocs for cPub : global symbol reference (2)

- the cPub symbol has the following characteristics
  - the symbol references data
  - the symbol is 1 byte long
- at run time, the dynamic linker will
  find the symbol cPub in one of the *libraries* and
  copy the 1 byte down from the library into the app data space
- the dynamic linker will then publish this new address
  as the address of cPub

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