

ELF1 7 Examples - 3 Executables - ELF Study 1999

Young W. Lim

2019-10-24 Thr

Outline

- 1 Based on
- 2 PIC relocs background
 - Relocs background in shared object and executable files
- 3 Executable reloc example using a shared library
 - Example codes
 - Compiling scripts
- 4 Relocations Results Summary
 - 1. rel.o object file relocations
 - 2. librel.so shared object file relocations
 - 3. main.o object file relocations
 - 4. run_dynamic executable file relocations
- 5 Relocations in a shared object (librel.so) file
 - TOC
 - Linking the .data section
 - Linking the .text section
- 6 Relocations in an executable (run_dynamic) file
 - TOC

Based on

"Study of ELF loading and relocs", 1999

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

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

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: Relocs background in shared object and executable files

- Relocs in a PIC shared object (.so) file
- Relocs in a non-PIC executable file
- PIC, PIE, and non-PIC executables

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 PICT shared library (.so) file (2)

| | | |
|----------------|-----|--|
| R_386_JMP_SLOT | S | <ul style="list-style-type: none">• PICT reference to a function symbol• offset : a PLT entry location• <u>fill</u> the GOT entry with a function symbol address |
| R_386_GLOB_DAT | S | <ul style="list-style-type: none">• PICT reference to a global symbol• offset to a GOT entry• <u>fill</u> the GOT entry with a global symbol address |
| R_386_RELATIVE | B+A | <ul style="list-style-type: none">• PICT reference to a local symbol• offset to a section• <u>add</u> 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 | <ul style="list-style-type: none">• <i>non-PIC</i> 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 | <ul style="list-style-type: none">• <i>PIC</i> reference to a global symbol• offset : a PLT entry location of a <i>PIC</i> 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

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/path/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/pathb/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>

Script 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
```

Script 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
```

Script 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
```

Script 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
```

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 | R_386_GOT32x | R_386_GOT32x |
| | absolute | GOT | GOT |
| fPub | R_386_PC32 | R_386_PC32 | R_386_PLT32 |
| | relative | relative | 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`

| | <code>-fno-pic</code> | <code>default</code> | <code>-fPIC</code> |
|-------------------|-----------------------------|-----------------------------|------------------------------|
| <code>fPub</code> | | | <code>R_386_JUMP_SLOT</code> |
| | <code>not applicable</code> | <code>not applicable</code> | <code>PLT</code> |

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 | R_386_JUMP_SLOT | R_386_JUMP_SLOT |
| | PLT | PLT | PLT |

4.g .rel.dyn relocs of run_dynamic

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

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           typedef struct {
                           fLocal }, // 2             char* p;
                           { &cPub, // 3             char (*f)(int);
                           fPub } }; // 4           } _st;
```

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

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 relocations turn into 4 relocations 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