CS451 – Software Analysis

Lecture 6

libelf

Elias Athanasopoulos
athanasopoulos.elias@ucy.ac.cy

libelf

- A more modern library that simplifies ELF parsing and loading
- Compared to libbfd, libelf focuses on the ELF format
- Useful for building ELF loaders
  - For instance, if you want to create a custom disassembler you need to load the ELF file, parse the sections, and process the code sections

`readelf –S` based on libelf

- readelf does not use libelf
- We will create a similar tool based on libelf
  - A tool that prints the sections of a binary
  - In addition, the tool will print the symbol table for non stripped binaries
  - Not the entire readelf functionality

Example of operation

```
$ readelf -SW /bin/ls
There are 31 section headers, starting at offset 0x22848:

Section Headers:
  [Nr] Name              Type            Address          Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            0000000000000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        0000000000000270 000270 00001c 00   A  0   0  1
  [ 2] .note.gnu.property NOTE           0000000000000290 000290 000020 00   A  0   0  8
  [ 3] .note.ABI-tag     NOTE            00000000000002b0 0002b0 000020 00   A  0   0  4
  [ 4] .note.gnu.build-id NOTE           00000000000002d0 0002d0 000024 00   A  0   0  4
[...]
```

Explanation

- Standard
  - Name, Address, Off (offset), Size
- Type
  - Different sections hold different types of data
  - PROGBITS is program’s data, STRTAB is ”strings table”, SYMTAB is “symbol table”, etc.
- Flg (Flags)
  - W (write), A (alloc), X (execute), M (merge), S (strings), I (info), L (link order), O (extra OS processing required), G (group), T (TLS), C (compressed), x (unknown), o (OS specific), E (exclude), l (large), p (processor specific)
- Other
  - ES (entry size if section holds table), Lk (link to another section), Inf (Additional section information), Al (section alignment)

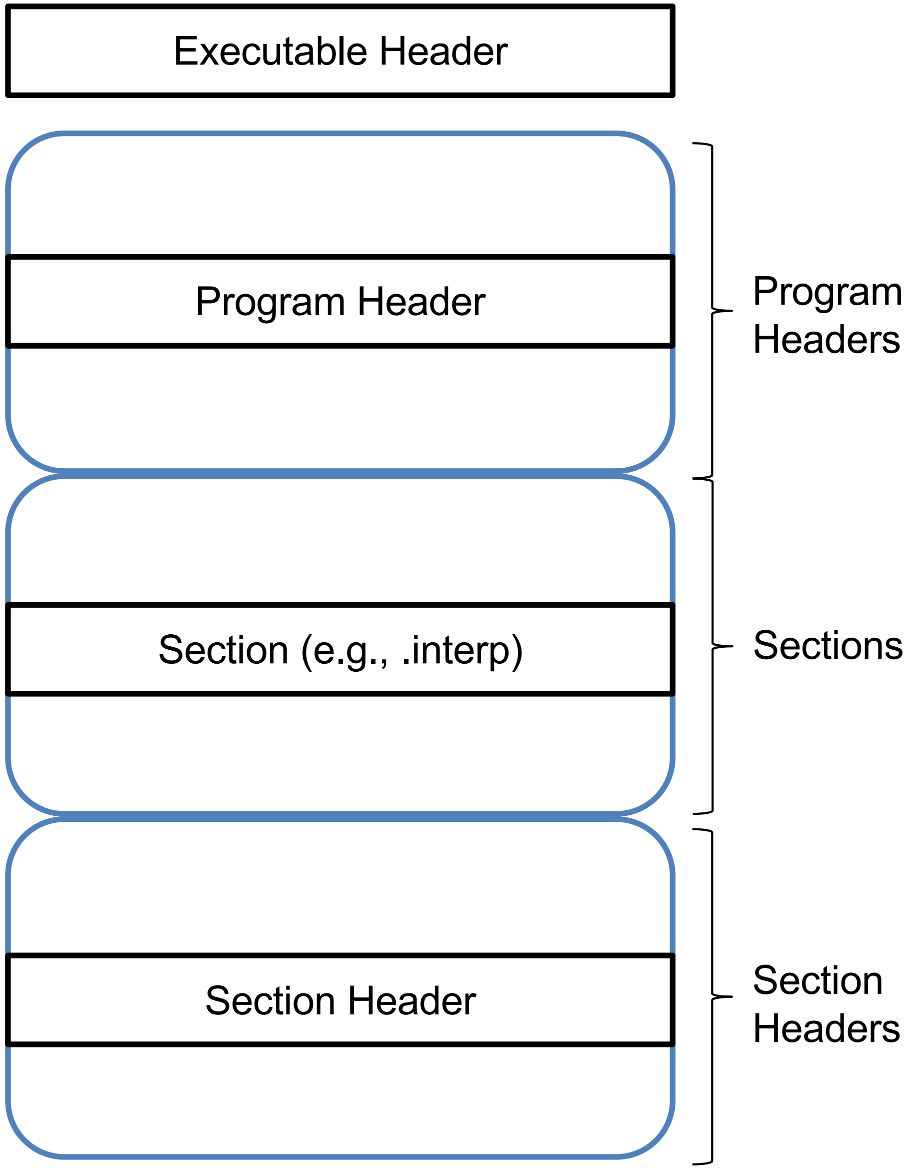
Example of operation

```
$ nm ./example-libelf
0000000000400485 t .annobin__dl_relocate_static_pie.end
0000000000400480 t .annobin__dl_relocate_static_pie.start
0000000000400560 t .annobin_elf_init.c
[…]
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
00000000004005d0 T __libc_csu_fini
0000000000400560 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
0000000000400541 T main
00000000004004c0 t register_tm_clones
0000000000400450 T _start
0000000000601020 D __TMC_END__
```

Explanation

- First column is the value of the symbol
- Second column is the type of the symbol
- Third column is the name of the symbol
- Columns may be empty

ELF Format



- Division is used by linkers
- Sections contain data but handling each section is done through their section header
- The section headers can be found through the Executable Header

Preliminaries

```
$ yum list installed | grep libelf
elfutils-libelf.x86_64                   0.185-1.el8     @baseos
elfutils-libelf-devel.x86_64    0.185-1.el8     @baseos
```

Requirements

```
$ yum list installed | grep binutils
binutils.x86_64                          2.30-108.el8_5.1
@baseos
binutils-devel.x86_64   2.30-108.el8_5.1 @appstream
```

- Useful files
  - /usr/include/elf.h
  - /usr/include/gelf.h
  - /usr/include/libelf.h

Compilation and run

```
$ gcc -Wall elfloader.c -lelf -o elfloader
$ ./elfloader ./elfloader-example
[ 0] .interp                    1 0000000000400238 000238 00001c
[ 1] .note.ABI-tag              7 0000000000400254 000254 000020
[ 2] .note.gnu.build-id         7 0000000000400274 000274 000024
[ 3] .gnu.hash         1879048182 0000000000400298 000298 00001c
[ 4] .dynsym                   11 00000000004002b8 0002b8 000078
[...]
[24] .symtab                    2 0000000000000000 002c08 000960
[25] .strtab                    3 0000000000000000 003568 000639
[26] .shstrtab                  3 0000000000000000 003ba1 00010f
Printing symbol table.
0000000000000000 0
000000000040047f 0 .annobin_init.c
000000000040047f 0 .annobin_init.c_end
0000000000400450 0 .annobin_init.c.hot
[...]
0000000000400541 12 main
0000000000601020 11 __TMC_END__
0000000000000000 20 _ITM_registerTMCloneTable
0000000000400428 12 _init
```

libelf Initialization

```
Elf *elf;

/* Initialization.  */
if (elf_version(EV_CURRENT) == EV_NONE)
    DIE("(version) %s", elf_errmsg(-1));

int fd = open(filename, O_RDONLY);

elf = elf_begin(fd, ELF_C_READ, NULL);
if (!elf)
    DIE("(begin) %s", elf_errmsg(-1));
```

Loop over sections

```
    Elf_Scn *scn = NULL;
    GElf_Shdr shdr;
    size_t shstrndx;
    if (elf_getshdrstrndx(elf, &shstrndx) != 0)
        DIE("(getshdrstrndx) %s", elf_errmsg(-1));

    int s_index = 0;
    while ((scn = elf_nextscn(elf, scn)) != NULL) {
        if (gelf_getshdr(scn, &shdr) != &shdr)
            DIE("(getshdr) %s", elf_errmsg(-1));

         [...]
    }
```

Print details for a section

```
  fprintf(stderr, "[%2d] %-20s %4d %016lx %06lx %06lx\n",
                   s_index++,
                   elf_strptr(elf, shstrndx, shdr.sh_name),
                   shdr.sh_type,
                   shdr.sh_addr,
                   shdr.sh_offset,
                   shdr.sh_size);
```

Locate the symbol table

```
        /* Locate symbol table.   */
        if (!strcmp(elf_strptr(elf, shstrndx, shdr.sh_name),
                 ".symtab"))
            symtab = scn;

} /* End of loop */

print_symbol_table(elf, symtab);
```

Handle the symbol table

```
void print_symbol_table(Elf *elf, Elf_Scn *scn) {
    Elf_Data *data;
    GElf_Shdr shdr;
    int count = 0;

    /* Get the descriptor.  */
    if (gelf_getshdr(scn, &shdr) != &shdr)
        DIE("(getshdr) %s", elf_errmsg(-1));

    data = elf_getdata(scn, NULL);
    count = shdr.sh_size / shdr.sh_entsize;

    fprintf(stderr, "Printing symbol table.\n");
    for (int i = 0; i < count; ++i) {
         /* print */
    }
}
```

Print symbols

```
GElf_Sym sym;
gelf_getsym(data, i, &sym);
if (ELF64_ST_TYPE(sym.st_info) == STT_FUNC ||
    ELF64_ST_TYPE(sym.st_info) == STT_OBJECT)
         fprintf(stderr, "%016lx %x %s\n",
                 sym.st_value,
                 sym.st_info,
                 elf_strptr(elf, shdr.sh_link, sym.st_name));
```

Homework

- Beautify the output of elfloader.c
- Use it with different binaries
  - You may spot bugs!