Федеральное агентство связи
Сибирский государственный университет телекоммуникаций и информатики

Лабораторная работа №6

Выполнил: студент группы ИП-211
Оганесян Альберт
Лацук Андрей
Проверил:
Профессор кафедры ПМиК
Малков Е. А.

Новосибирск 2024

Задание: получите список *имён* экспортируемых функций библиотеки совместного доступа, разработанной в лабораторной 5. Дополните программу лекции 6 алгоритмом поиска имён разделов.
Подсказка: используйте поле заголовка ELF файла e_shstrndx и раздел .shstrtab.
Цель: знакомство со структурой ELF файлов.

Ход работы:

1. При помощи команды readelf –Ws liblist.so | grep FUNC получаем список экспортируемых функций (Флаг -W отключает предупреждения. Флаг -s показывает таблицу символов файла. При помощи grep FUNC выбираем только строки, содержащие “FUNC”)

```
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/6$ readelf -Ws liblist.so | grep FUNC
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND free@GLIBC_2.2.5 (2)
     2: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND putchar@GLIBC_2.2.5 (2)
     3: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND strncpy@GLIBC_2.2.5 (2)
     5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND puts@GLIBC_2.2.5 (2)
     6: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND printf@GLIBC_2.2.5 (2)
     8: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND memcpy@GLIBC_2.14 (3)
     9: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND malloc@GLIBC_2.2.5 (2)
    10: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND exit@GLIBC_2.2.5 (2)
    12: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND __cxa_finalize@GLIBC_2.2.5 (2)
    13: 0000000000001179   164 FUNC    GLOBAL DEFAULT   11 createStudent
    14: 0000000000001249   201 FUNC    GLOBAL DEFAULT   11 printStudents
    15: 000000000000121d    44 FUNC    GLOBAL DEFAULT   11 addStudent
    16: 0000000000001312    67 FUNC    GLOBAL DEFAULT   11 freeStudents
    26: 00000000000010c0     0 FUNC    LOCAL  DEFAULT   11 deregister_tm_clones
    27: 00000000000010f0     0 FUNC    LOCAL  DEFAULT   11 register_tm_clones
    28: 0000000000001130     0 FUNC    LOCAL  DEFAULT   11 __do_global_dtors_aux
    31: 0000000000001170     0 FUNC    LOCAL  DEFAULT   11 frame_dummy
    37: 0000000000001358     0 FUNC    LOCAL  DEFAULT   12 _fini
    43: 0000000000001000     0 FUNC    LOCAL  DEFAULT    9 _init
    44: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND free@@GLIBC_2.2.5
    45: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND putchar@@GLIBC_2.2.5
    46: 0000000000001179   164 FUNC    GLOBAL DEFAULT   11 createStudent
    47: 0000000000001249   201 FUNC    GLOBAL DEFAULT   11 printStudents
    48: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND strncpy@@GLIBC_2.2.5
    50: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND puts@@GLIBC_2.2.5
    51: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND printf@@GLIBC_2.2.5
    53: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND memcpy@@GLIBC_2.14
    54: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND malloc@@GLIBC_2.2.5
    55: 000000000000121d    44 FUNC    GLOBAL DEFAULT   11 addStudent
    56: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND exit@@GLIBC_2.2.5
    58: 0000000000001312    67 FUNC    GLOBAL DEFAULT   11 freeStudents
    59: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND __cxa_finalize@@GLIBC_2.2.5
```

Рис. 1.1 список экспортируемых функций

2. Доработаем программу из лекции поиском имен разделов в ELF файле:

```
#include <elf.h>

#include <stdio.h>
```

```
#include <string.h>
#include <stdlib.h>
int main(int argc, char** argv){
  const char* elfFile=argv[1];
  Elf64_Ehdr header;
  Elf64_Shdr sheader;
  Elf64_Shdr symtab;
  Elf64_Shdr strtab;
  Elf64_Shdr shstrtab;
  Elf64_Sym sym;
  char sname[32];
  char sectionName[32];
  int i;
  FILE* file = fopen(elfFile, "rb");
  if(file==NULL){
    fprintf(stderr, "Error opening file %s\n", elfFile);
    return 1;
  }
  fread(&header, sizeof(header), 1, file);

   fseek(file, header.e_shoff, SEEK_SET);

  fread(&sheader, sizeof(sheader), 1, file);
  for(i=0; i<header.e_shnum;i++){
```

```
  fseek(file,header.e_shoff+header.e_shentsize*i, SEEK_SET);

  fread(&sheader, sizeof(sheader), 1, file);
  if(i==4)
    symtab=(Elf64_Shdr)sheader;
  if(i==5)
   strtab=(Elf64_Shdr)sheader;
}

fseek(file, header.e_shoff + header.e_shentsize * header.e_shstrndx, SEEK_SET);
fread(&shstrtab, sizeof(shstrtab), 1, file);

printf("Section Names:\n");
for (i = 0; i < header.e_shnum; i++) {
    fseek(file, header.e_shoff + header.e_shentsize * i, SEEK_SET);
    fread(&sheader, sizeof(sheader), 1, file);

    fseek(file, shstrtab.sh_offset + sheader.sh_name, SEEK_SET);
    fread(sectionName, 1, sizeof(sectionName) - 1, file);
    sectionName[31] = '\0';

    printf("Section %d: %s\n", i, sectionName);
}
```

```
for (i = 0; i < header.e_shnum; i++) {
  fseek(file, header.e_shoff + header.e_shentsize * i, SEEK_SET);
  if (fread(&sheader, sizeof(sheader), 1, file) != 1) {
    fprintf(stderr, "Failed to read section header %d\n", i);
    fclose(file);
    return 1;
  }

  fseek(file, shstrtab.sh_offset + sheader.sh_name, SEEK_SET);
  fread(sectionName, 1, sizeof(sectionName) - 1, file);
  sectionName[31] = '\0';

  if (strcmp(sectionName, ".symtab") == 0) {
    symtab = sheader;
  } else if (strcmp(sectionName, ".strtab") == 0) {
    strtab = sheader;
  }
}

for(i=0;i<symtab.sh_size / symtab.sh_entsize;i++)
  {
   fseek(file,symtab.sh_offset + symtab.sh_entsize*i, SEEK_SET);
   fread(&sym, sizeof(Elf64_Sym), 1, file);
```

```
    fseek(file,strtab.sh_offset+sym.st_name, SEEK_SET);
    fread(sname, 1,32, file);
    fprintf(stdout, "%d\t%lld\t%u\t%u\t%hd\t%s\n", i,
            sym.st_size,
            ELF64_ST_TYPE(sym.st_info),
            ELF64_ST_BIND(sym.st_info),
            sym.st_shndx, sname);
   }


  return 0;
}
```

3. Скомпилируем и запустим программу на примере .so файла из 5-й лабораторной:

```
albert@DESKTOP-70OAJI4:/mnt/c/Users/User/Documents/GitHub/OS/6$ ./a.out liblist.so
Section Names:
Section 0:
Section 1: .note.gnu.build-id
Section 2: .gnu.hash
Section 3: .dynsym
Section 4: .dynstr
Section 5: .gnu.version
Section 6: .gnu.version_r
Section 7: .rela.dyn
Section 8: .rela.plt
Section 9: .init
Section 10: .plt
Section 11: .text
Section 12: .fini
Section 13: .rodata
Section 14: .eh_frame_hdr
Section 15: .eh_frame
Section 16: .init_array
Section 17: .fini_array
Section 18: .data.rel.ro
Section 19: .dynamic
Section 20: .got
Section 21: .got.plt
Section 22: .bss
Section 23: .comment
Section 24: .gnu.build.attributes
Section 25: .symtab
Section 26: .strtab
Section 27: .shstrtab
```

Рис. 3.1 Вывод найденных разделов

Вывод: мы познакомились со структурой ELF файлов и научились получать из него информацию