МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Липецкий государственный технический университет
Факультет автоматизации и информатики

Лабораторная работа №2
по операционным системам
“Исследование исполняемых модулей”

Студент Станиславчук С. М.
Группа АС-21-1

Руководитель Останков А. И.

Липецк 2023 г.

2. Цель работы, задание

Цель работы:

Изучить содержимое исполняемых файлов.

Задание:

1. Установить среду программирования gcc

2. Разработать небольшую программу на языке с (20-30 строк). В программе должны присутствовать:

- массивы или строки с заданными исходными данными
- циклы обработки массивов с вычислением результата
- условные операторы в процессе вычисления
- вывод результатов расчётов в stdout

Темы программы не должны повторяться у разных студентов, поэтому нужно её согласовать друг с другом и/или с преподавателем.

3. Необходимо отладить программу. Т.е. добиться её корректного исполнения.

4. Получить три варианта модуля программы:

- объектный файл (результат cc -c <file.c>)
- исполняемый файл, собранный статически (результат cc -static <file.c>)
- исполняемый файл, собранный динамически (результат cc <file.c>)

В двух последних случаях целесообразно использовать ключ -o для указания имени создаваемого модуля

5. Сравнить три полученных модуля одной и той же программы. по размеру

6. Исследовать содержимое трех полученных файлов с помощью команд objdump и readelf и объяснить:

- состав сегментов, включенных в исполняемые модули, какова их роль и как они располагаются в памяти
- состав программных секций, размещённых в сегментах и откуда они появились в исполняемых модулях
- таблицу символов

7. Идентифицировать секцию, содержащую код программы и попытаться его дизассемблировать. Интерпретировать полученный результат

8. Объяснить разницу в размерах трех модулей.

Отчёт должен содержать:

- исходный текст программы
- её дизассемблированный вариант
- таблицы сегментов (для двух исполняемых модулей)
- таблицы секций (в трёх вариантах)

Код программы:

```
#include <stdio.h>
const int arraySize = 11;
const int residual = 8;
float power(float value, int power){
	float result = 1;
	while (power >  0){
		result *= value;
		power -= 1;
	}
	return result;
}
int main(){
	int array[11] = {1, 2, 4, 8, 16, 17, 25, 46, 14, 5, 17};
	int result = 0;
	float poweredValue = power(array[arraySize-1], arraySize);
	for (int i = 0; i < arraySize; i++)
	{
		float poweredValue = power(array[i], i);
		if (result % residual == 0)
		{
			printf("result that == division without residual == 0 = %d", result);
			printf("\n");
		}
		result = poweredValue;
	}
	return 0;
}
```

Дизассемблированный вариант:

Дизассемблированная секция кода командой **objdump -d --section=.text script.o**:

```
0000000000000000 <power>:
   0:   f3 0f 1e fa           endbr64
   4:   55                    push   %rbp
   5:   48 89 e5              mov    %rsp,%rbp
   8:   f3 0f 11 45 ec        movss  %xmm0,-0x14(%rbp)
   d:   89 7d e8              mov    %edi,-0x18(%rbp)
  10:   f3 0f 10 05 00 00 00  movss  0x0(%rip),%xmm0        # 18 <power+0x18>
  17:   00
  18:   f3 0f 11 45 fc        movss  %xmm0,-0x4(%rbp)
  1d:   eb 13                 jmp    32 <power+0x32>
  1f:   f3 0f 10 45 fc        movss  -0x4(%rbp),%xmm0
  24:   f3 0f 59 45 ec        mulss  -0x14(%rbp),%xmm0
  29:   f3 0f 11 45 fc        movss  %xmm0,-0x4(%rbp)
  2e:   83 6d e8 01           subl   $0x1,-0x18(%rbp)
  32:   83 7d e8 00           cmpl   $0x0,-0x18(%rbp)
  36:   7f e7                 jg     1f <power+0x1f>
  38:   f3 0f 10 45 fc        movss  -0x4(%rbp),%xmm0
  3d:   5d                    pop    %rbp
  3e:   c3                    ret


000000000000003f <main>:
  3f:   f3 0f 1e fa           endbr64
  43:   55                    push   %rbp
  44:   48 89 e5              mov    %rsp,%rbp
  47:   48 83 ec 50           sub    $0x50,%rsp
  4b:   64 48 8b 04 25 28 00  mov    %fs:0x28,%rax
  52:   00 00
  54:   48 89 45 f8           mov    %rax,-0x8(%rbp)
  58:   31 c0                 xor    %eax,%eax
  5a:   c7 45 c0 01 00 00 00  movl   $0x1,-0x40(%rbp)
  61:   c7 45 c4 02 00 00 00  movl   $0x2,-0x3c(%rbp)
  68:   c7 45 c8 04 00 00 00  movl   $0x4,-0x38(%rbp)
  6f:   c7 45 cc 08 00 00 00  movl   $0x8,-0x34(%rbp)
  76:   c7 45 d0 10 00 00 00  movl   $0x10,-0x30(%rbp)
  7d:   c7 45 d4 11 00 00 00  movl   $0x11,-0x2c(%rbp)
  84:   c7 45 d8 19 00 00 00  movl   $0x19,-0x28(%rbp)
  8b:   c7 45 dc 2e 00 00 00  movl   $0x2e,-0x24(%rbp)
```

```
 92:   c7 45 e0 0e 00 00 00  movl   $0xe,-0x20(%rbp)
 99:   c7 45 e4 05 00 00 00  movl   $0x5,-0x1c(%rbp)
 a0:   c7 45 e8 11 00 00 00  movl   $0x11,-0x18(%rbp)
 a7:   c7 45 b0 01 00 00 00  movl   $0x1,-0x50(%rbp)
 ae:   ba 0b 00 00 00        mov    $0xb,%edx
 b3:   b8 0b 00 00 00        mov    $0xb,%eax
 b8:   83 e8 01              sub    $0x1,%eax
 bb:   48 98                 cltq
 bd:   8b 44 85 c0           mov    -0x40(%rbp,%rax,4),%eax
 c1:   66 0f ef c9           pxor   %xmm1,%xmm1
 c5:   f3 0f 2a c8           cvtsi2ss %eax,%xmm1
 c9:   66 0f 7e c8           movd   %xmm1,%eax
 cd:   89 d7                 mov    %edx,%edi
 cf:   66 0f 6e c0           movd   %eax,%xmm0
 d3:   e8 00 00 00 00        call   d8 <main+0x99>
 d8:   66 0f 7e c0           movd   %xmm0,%eax
 dc:   89 45 b8              mov    %eax,-0x48(%rbp)
 df:   c7 45 b4 00 00 00 00  movl   $0x0,-0x4c(%rbp)
 e6:   eb 6e                 jmp    156 <main+0x117>
 e8:   8b 45 b4              mov    -0x4c(%rbp),%eax
 eb:   48 98                 cltq
 ed:   8b 44 85 c0           mov    -0x40(%rbp,%rax,4),%eax
 f1:   66 0f ef d2           pxor   %xmm2,%xmm2
 f5:   f3 0f 2a d0           cvtsi2ss %eax,%xmm2
 f9:   66 0f 7e d0           movd   %xmm2,%eax
 fd:   8b 55 b4              mov    -0x4c(%rbp),%edx
100:   89 d7                 mov    %edx,%edi
102:   66 0f 6e c0           movd   %eax,%xmm0
106:   e8 00 00 00 00        call   10b <main+0xcc>
10b:   66 0f 7e c0           movd   %xmm0,%eax
10f:   89 45 bc              mov    %eax,-0x44(%rbp)
112:   b9 08 00 00 00        mov    $0x8,%ecx
117:   8b 45 b0              mov    -0x50(%rbp),%eax
11a:   99                    cltd
11b:   f7 f9                 idiv   %ecx
11d:   89 d0                 mov    %edx,%eax
11f:   85 c0                 test   %eax,%eax
121:   75 23                 jne    146 <main+0x107>
123:   8b 45 b0              mov    -0x50(%rbp),%eax
126:   89 c6                 mov    %eax,%esi
128:   48 8d 05 00 00 00 00  lea    0x0(%rip),%rax        # 12f <main+0xf0>
```

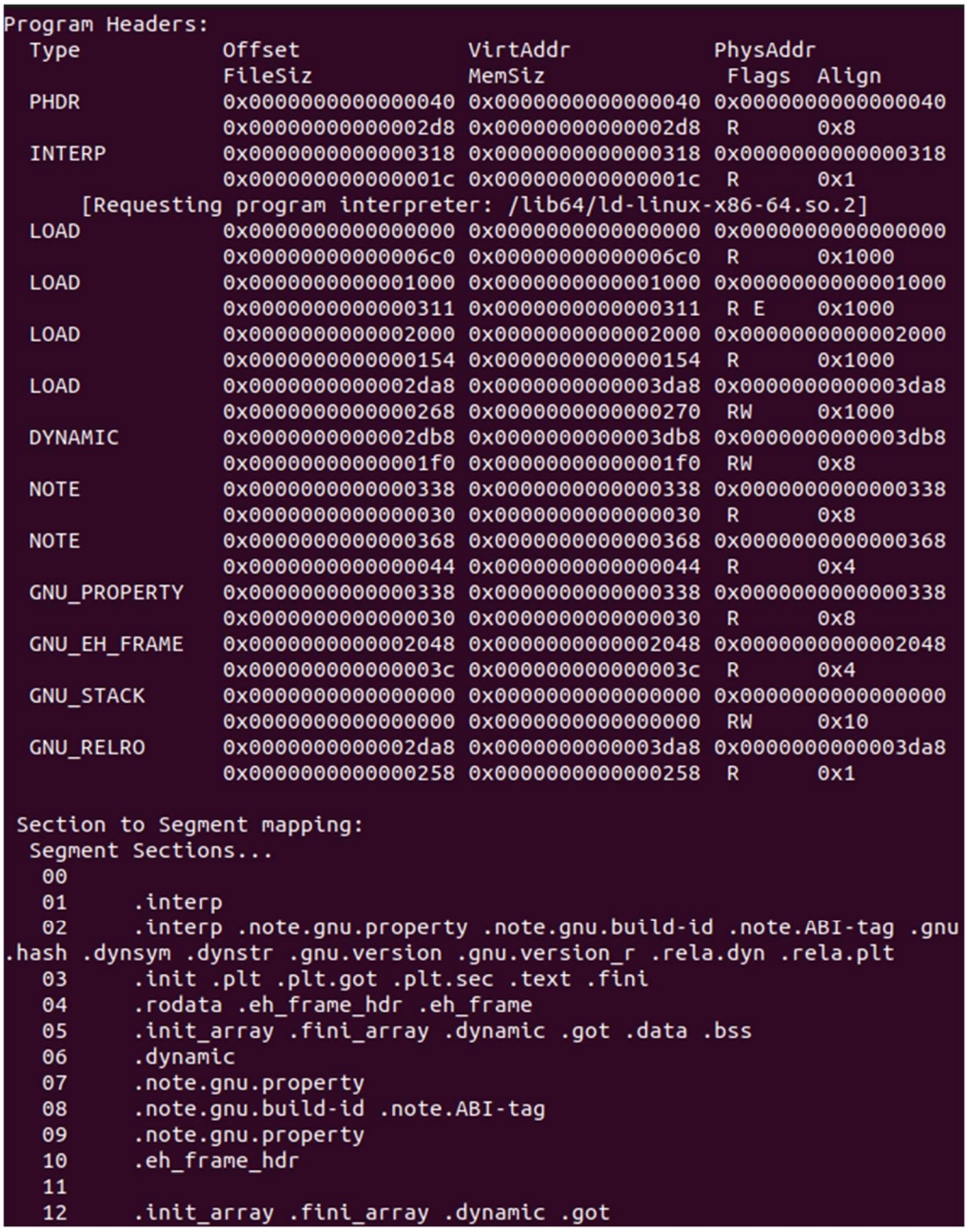
```
12f:   48 89 c7              mov    %rax,%rdi
132:   b8 00 00 00 00        mov    $0x0,%eax
137:   e8 00 00 00 00        call   13c <main+0xfd>
13c:   bf 0a 00 00 00        mov    $0xa,%edi
141:   e8 00 00 00 00        call   146 <main+0x107>
146:   f3 0f 10 45 bc        movss  -0x44(%rbp),%xmm0
14b:   f3 0f 2c c0           cvttss2si %xmm0,%eax
14f:   89 45 b0              mov    %eax,-0x50(%rbp)
152:   83 45 b4 01           addl   $0x1,-0x4c(%rbp)
156:   b8 0b 00 00 00        mov    $0xb,%eax
15b:   39 45 b4              cmp    %eax,-0x4c(%rbp)
15e:   7c 88                 jl     e8 <main+0xa9>
160:   b8 00 00 00 00        mov    $0x0,%eax
165:   48 8b 55 f8           mov    -0x8(%rbp),%rdx
169:   64 48 2b 14 25 28 00  sub    %fs:0x28,%rdx
170:   00 00
172:   74 05                 je     179 <main+0x13a>
174:   e8 00 00 00 00        call   179 <main+0x13a>
179:   c9                    leave
17a:   c3                    ret
```

Таблицы сегментов:

Для отображения таблицы сегментов используется команда **readelf -l script_name**

Для отображения таблицы программных секций используется команда **readelf -S script_name**

Для отображения таблицы символов используется команда **readelf -s script_name**



```
Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000000040 0x0000000000000040
                 0x00000000000002d8 0x00000000000002d8  R      0x8
  INTERP         0x0000000000000318 0x0000000000000318 0x0000000000000318
                 0x000000000000001c 0x000000000000001c  R      0x1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x00000000000006c0 0x00000000000006c0  R      0x1000
  LOAD           0x0000000000001000 0x0000000000001000 0x0000000000001000
                 0x0000000000000311 0x0000000000000311  R E    0x1000
  LOAD           0x0000000000002000 0x0000000000002000 0x0000000000002000
                 0x0000000000000154 0x0000000000000154  R      0x1000
  LOAD           0x0000000000002da8 0x0000000000003da8 0x0000000000003da8
                 0x0000000000000268 0x0000000000000270  RW     0x1000
  DYNAMIC        0x0000000000002db8 0x0000000000003db8 0x0000000000003db8
                 0x00000000000001f0 0x00000000000001f0  RW     0x8
  NOTE           0x0000000000000338 0x0000000000000338 0x0000000000000338
                 0x0000000000000030 0x0000000000000030  R      0x8
  NOTE           0x0000000000000368 0x0000000000000368 0x0000000000000368
                 0x0000000000000044 0x0000000000000044  R      0x4
  GNU_PROPERTY   0x0000000000000338 0x0000000000000338 0x0000000000000338
                 0x0000000000000030 0x0000000000000030  R      0x8
  GNU_EH_FRAME   0x0000000000002048 0x0000000000002048 0x0000000000002048
                 0x000000000000003c 0x000000000000003c  R      0x4
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     0x10
  GNU_RELRO      0x0000000000002da8 0x0000000000003da8 0x0000000000003da8
                 0x0000000000000258 0x0000000000000258  R      0x1

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.gnu.property .note.gnu.build-id .note.ABI-tag .gnu
.hash .dynsym .dynstr .gnu.version .gnu.version_r .rela.dyn .rela.plt
   03     .init .plt .plt.got .plt.sec .text .fini
   04     .rodata .eh_frame_hdr .eh_frame
   05     .init_array .fini_array .dynamic .got .data .bss
   06     .dynamic
   07     .note.gnu.property
   08     .note.gnu.build-id .note.ABI-tag
   09     .note.gnu.property
   10     .eh_frame_hdr
   11
   12     .init_array .fini_array .dynamic .got
```

Рисунок 1 – таблица сегментов **script_dynamic**

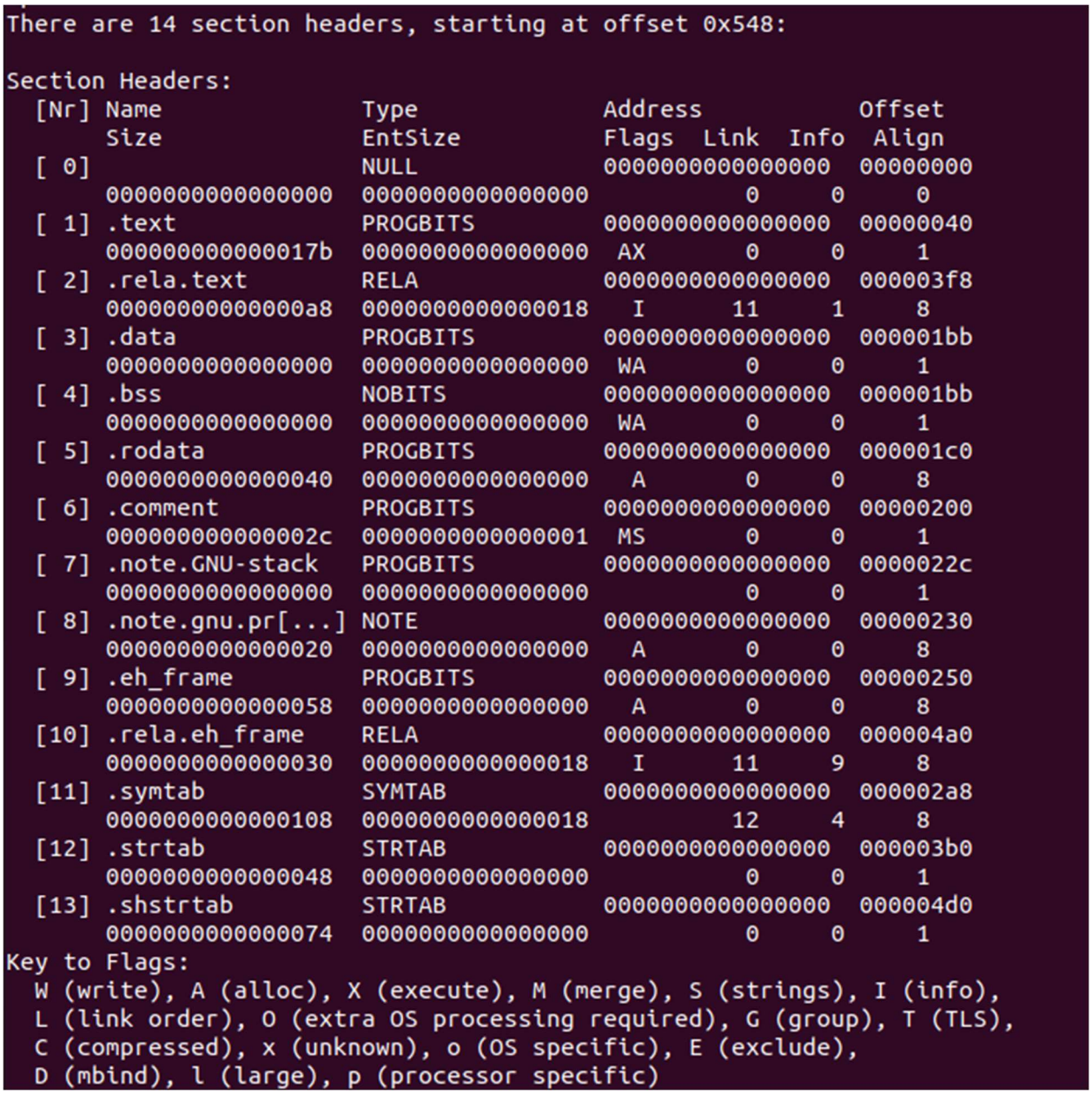
```
There are 10 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x0000000000000528 0x0000000000000528  R      0x1000
  LOAD           0x0000000000001000 0x0000000000401000 0x0000000000401000
                 0x00000000000967bd 0x00000000000967bd  R E    0x1000
  LOAD           0x0000000000098000 0x0000000000498000 0x0000000000498000
                 0x00000000000285e2 0x00000000000285e2  R      0x1000
  LOAD           0x00000000000c07b0 0x00000000004c17b0 0x00000000004c17b0
                 0x0000000000005ae0 0x000000000000b490  RW     0x1000
  NOTE           0x0000000000000270 0x0000000000400270 0x0000000000400270
                 0x0000000000000030 0x0000000000000030  R      0x8
  NOTE           0x00000000000002a0 0x00000000004002a0 0x00000000004002a0
                 0x0000000000000044 0x0000000000000044  R      0x4
  TLS            0x00000000000c07b0 0x00000000004c17b0 0x00000000004c17b0
                 0x0000000000000020 0x0000000000000068  R      0x8
  GNU_PROPERTY   0x0000000000000270 0x0000000000400270 0x0000000000400270
                 0x0000000000000030 0x0000000000000030  R      0x8
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     0x10
  GNU_RELRO      0x00000000000c07b0 0x00000000004c17b0 0x00000000004c17b0
                 0x0000000000003850 0x0000000000003850  R      0x1

 Section to Segment mapping:
  Segment Sections...
   00     .note.gnu.property .note.gnu.build-id .note.ABI-tag .rela.plt
   01     .init .plt .text __libc_freeres_fn .fini
   02     .rodata .stapsdt.base .eh_frame .gcc_except_table
   03     .tdata .init_array .fini_array .data.rel.ro .got .got.plt .data __
libc_subfreeres __libc_IO_vtables __libc_atexit .bss __libc_freeres_ptrs
   04     .note.gnu.property
   05     .note.gnu.build-id .note.ABI-tag
   06     .tdata .tbss
   07     .note.gnu.property
   08
   09     .tdata .init_array .fini_array .data.rel.ro .got
```

Рисунок 2 – таблица сегментов **script_static**

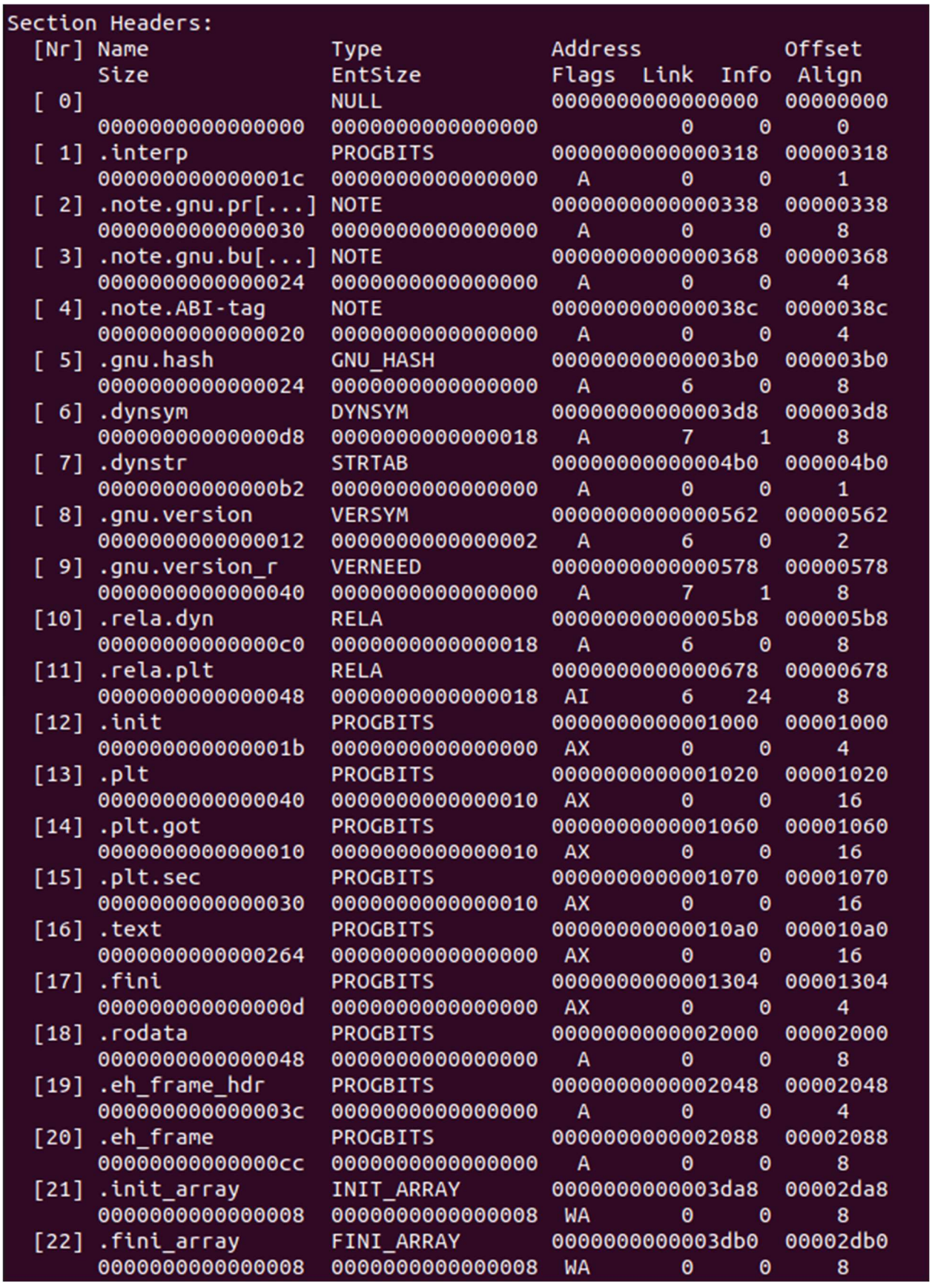
Таблицы программных секций:



```
There are 14 section headers, starting at offset 0x548:

Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .text             PROGBITS         0000000000000000  00000040
       000000000000017b  0000000000000000  AX       0     0     1
  [ 2] .rela.text        RELA             0000000000000000  000003f8
       00000000000000a8  0000000000000018   I      11     1     8
  [ 3] .data             PROGBITS         0000000000000000  000001bb
       0000000000000000  0000000000000000  WA       0     0     1
  [ 4] .bss              NOBITS           0000000000000000  000001bb
       0000000000000000  0000000000000000  WA       0     0     1
  [ 5] .rodata           PROGBITS         0000000000000000  000001c0
       0000000000000040  0000000000000000   A       0     0     8
  [ 6] .comment          PROGBITS         0000000000000000  00000200
       000000000000002c  0000000000000001  MS       0     0     1
  [ 7] .note.GNU-stack   PROGBITS         0000000000000000  0000022c
       0000000000000000  0000000000000000           0     0     1
  [ 8] .note.gnu.pr[...] NOTE             0000000000000000  00000230
       0000000000000020  0000000000000000   A       0     0     8
  [ 9] .eh_frame         PROGBITS         0000000000000000  00000250
       0000000000000058  0000000000000000   A       0     0     8
  [10] .rela.eh_frame    RELA             0000000000000000  000004a0
       0000000000000030  0000000000000018   I      11     9     8
  [11] .symtab           SYMTAB           0000000000000000  000002a8
       0000000000000108  0000000000000018          12     4     8
  [12] .strtab           STRTAB           0000000000000000  000003b0
       0000000000000048  0000000000000000           0     0     1
  [13] .shstrtab         STRTAB           0000000000000000  000004d0
       0000000000000074  0000000000000000           0     0     1
Key to 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),
  D (mbind), l (large), p (processor specific)
```

Рисунок 3 - Таблица программных секций **script.o**



```
Section Headers:
  [Nr] Name              Type            Address           Offset
       Size              EntSize         Flags  Link  Info  Align
  [ 0]                   NULL            0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .interp           PROGBITS        0000000000000318  00000318
       000000000000001c  0000000000000000   A       0     0     1
  [ 2] .note.gnu.pr[...] NOTE            0000000000000338  00000338
       0000000000000030  0000000000000000   A       0     0     8
  [ 3] .note.gnu.bu[...] NOTE            0000000000000368  00000368
       0000000000000024  0000000000000000   A       0     0     4
  [ 4] .note.ABI-tag     NOTE            000000000000038c  0000038c
       0000000000000020  0000000000000000   A       0     0     4
  [ 5] .gnu.hash         GNU_HASH        00000000000003b0  000003b0
       0000000000000024  0000000000000000   A       6     0     8
  [ 6] .dynsym           DYNSYM          00000000000003d8  000003d8
       00000000000000d8  0000000000000018   A       7     1     8
  [ 7] .dynstr           STRTAB          00000000000004b0  000004b0
       00000000000000b2  0000000000000000   A       0     0     1
  [ 8] .gnu.version      VERSYM          0000000000000562  00000562
       0000000000000012  0000000000000002   A       6     0     2
  [ 9] .gnu.version_r    VERNEED         0000000000000578  00000578
       0000000000000040  0000000000000000   A       7     1     8
  [10] .rela.dyn         RELA            00000000000005b8  000005b8
       00000000000000c0  0000000000000018   A       6     0     8
  [11] .rela.plt         RELA            0000000000000678  00000678
       0000000000000048  0000000000000018  AI       6    24     8
  [12] .init             PROGBITS        0000000000001000  00001000
       000000000000001b  0000000000000000  AX       0     0     4
  [13] .plt              PROGBITS        0000000000001020  00001020
       0000000000000040  0000000000000010  AX       0     0     16
  [14] .plt.got          PROGBITS        0000000000001060  00001060
       0000000000000010  0000000000000010  AX       0     0     16
  [15] .plt.sec          PROGBITS        0000000000001070  00001070
       0000000000000030  0000000000000010  AX       0     0     16
  [16] .text             PROGBITS        00000000000010a0  000010a0
       0000000000000264  0000000000000000  AX       0     0     16
  [17] .fini             PROGBITS        0000000000001304  00001304
       000000000000000d  0000000000000000  AX       0     0     4
  [18] .rodata           PROGBITS        0000000000002000  00002000
       0000000000000048  0000000000000000   A       0     0     8
  [19] .eh_frame_hdr     PROGBITS        0000000000002048  00002048
       000000000000003c  0000000000000000   A       0     0     4
  [20] .eh_frame         PROGBITS        0000000000002088  00002088
       00000000000000cc  0000000000000000   A       0     0     8
  [21] .init_array       INIT_ARRAY      0000000000003da8  00002da8
       0000000000000008  0000000000000008  WA       0     0     8
  [22] .fini_array       FINI_ARRAY      0000000000003db0  00002db0
       0000000000000008  0000000000000008  WA       0     0     8
```

Рисунок 4 – Таблица программных секций **script_dynamic** (0 – 22)

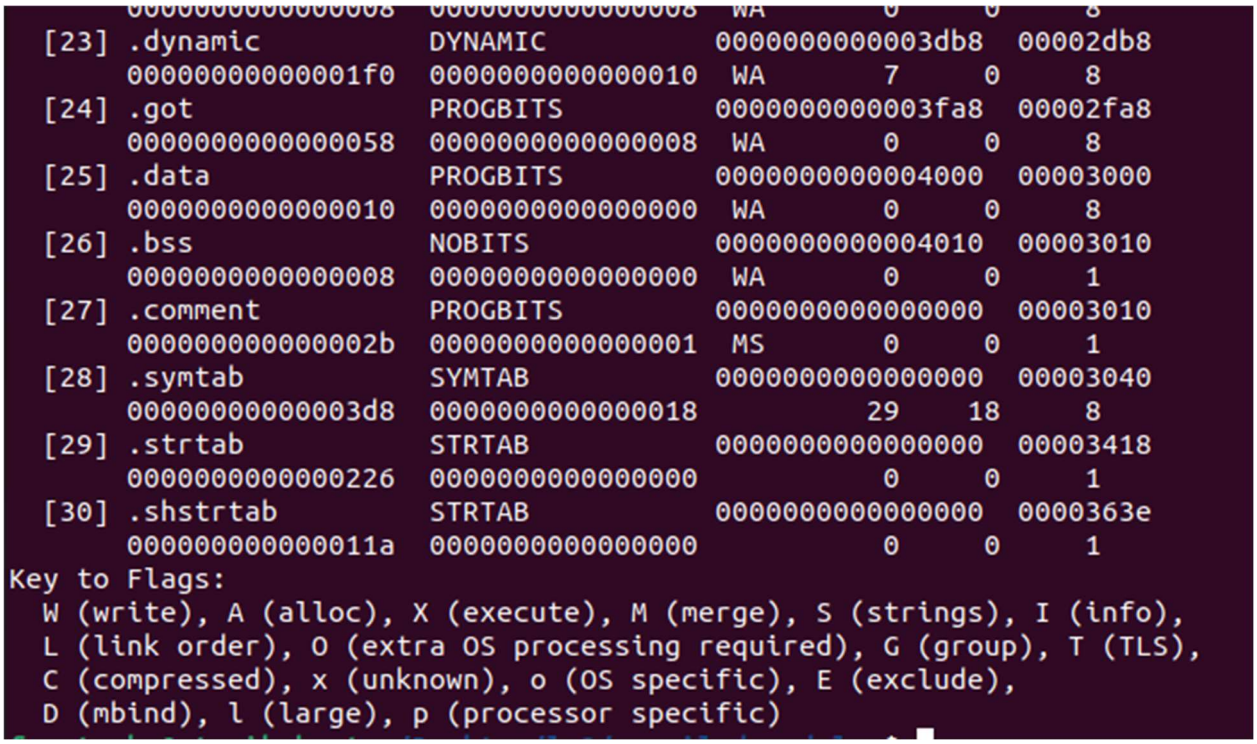
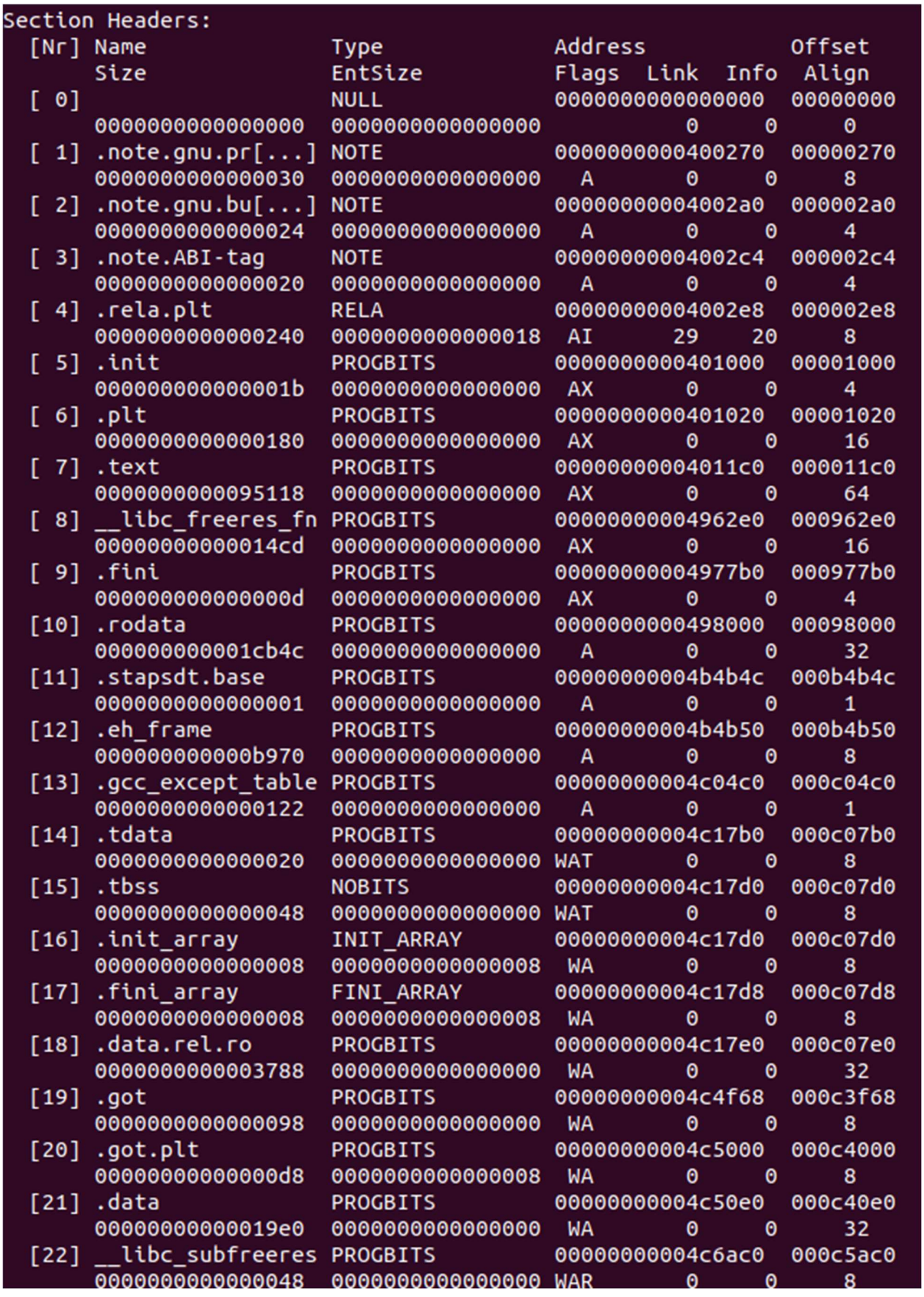


Рисунок 5 – Таблица программных секций **script_dynamic** (23 – 30)



```
Section Headers:
  [Nr] Name              Type            Address           Offset
       Size              EntSize         Flags  Link  Info  Align
  [ 0]                   NULL            0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .note.gnu.pr[...] NOTE            0000000000400270  00000270
       0000000000000030  0000000000000000   A       0     0     8
  [ 2] .note.gnu.bu[...] NOTE            00000000004002a0  000002a0
       0000000000000024  0000000000000000   A       0     0     4
  [ 3] .note.ABI-tag     NOTE            00000000004002c4  000002c4
       0000000000000020  0000000000000000   A       0     0     4
  [ 4] .rela.plt         RELA            00000000004002e8  000002e8
       0000000000000240  0000000000000018  AI      29    20     8
  [ 5] .init             PROGBITS        0000000000401000  00001000
       000000000000001b  0000000000000000  AX       0     0     4
  [ 6] .plt              PROGBITS        0000000000401020  00001020
       0000000000000180  0000000000000000  AX       0     0     16
  [ 7] .text             PROGBITS        00000000004011c0  000011c0
       0000000000095118  0000000000000000  AX       0     0     64
  [ 8] __libc_freeres_fn PROGBITS        00000000004962e0  000962e0
       00000000000014cd  0000000000000000  AX       0     0     16
  [ 9] .fini             PROGBITS        00000000004977b0  000977b0
       000000000000000d  0000000000000000  AX       0     0     4
  [10] .rodata           PROGBITS        0000000000498000  00098000
       000000000001cb4c  0000000000000000   A       0     0     32
  [11] .stapsdt.base     PROGBITS        00000000004b4b4c  000b4b4c
       0000000000000001  0000000000000000   A       0     0     1
  [12] .eh_frame         PROGBITS        00000000004b4b50  000b4b50
       000000000000b970  0000000000000000   A       0     0     8
  [13] .gcc_except_table PROGBITS        00000000004c04c0  000c04c0
       0000000000000122  0000000000000000   A       0     0     1
  [14] .tdata            PROGBITS        00000000004c17b0  000c07b0
       0000000000000020  0000000000000000 WAT       0     0     8
  [15] .tbss             NOBITS          00000000004c17d0  000c07d0
       0000000000000048  0000000000000000 WAT       0     0     8
  [16] .init_array       INIT_ARRAY      00000000004c17d0  000c07d0
       0000000000000008  0000000000000008  WA       0     0     8
  [17] .fini_array       FINI_ARRAY      00000000004c17d8  000c07d8
       0000000000000008  0000000000000008  WA       0     0     8
  [18] .data.rel.ro      PROGBITS        00000000004c17e0  000c07e0
       0000000000003788  0000000000000000  WA       0     0     32
  [19] .got              PROGBITS        00000000004c4f68  000c3f68
       0000000000000098  0000000000000000  WA       0     0     8
  [20] .got.plt          PROGBITS        00000000004c5000  000c4000
       00000000000000d8  0000000000000008  WA       0     0     8
  [21] .data             PROGBITS        00000000004c50e0  000c40e0
       00000000000019e0  0000000000000000  WA       0     0     32
  [22] __libc_subfreeres PROGBITS        00000000004c6ac0  000c5ac0
       0000000000000048  0000000000000000 WAR       0     0     8
```

Рисунок 6 – Таблица программных секций script_static (0 – 22)

Рисунок 7 – Таблица программных секций script_static (23 – 31)

Вывод: научился работать с терминалом Ubuntu.