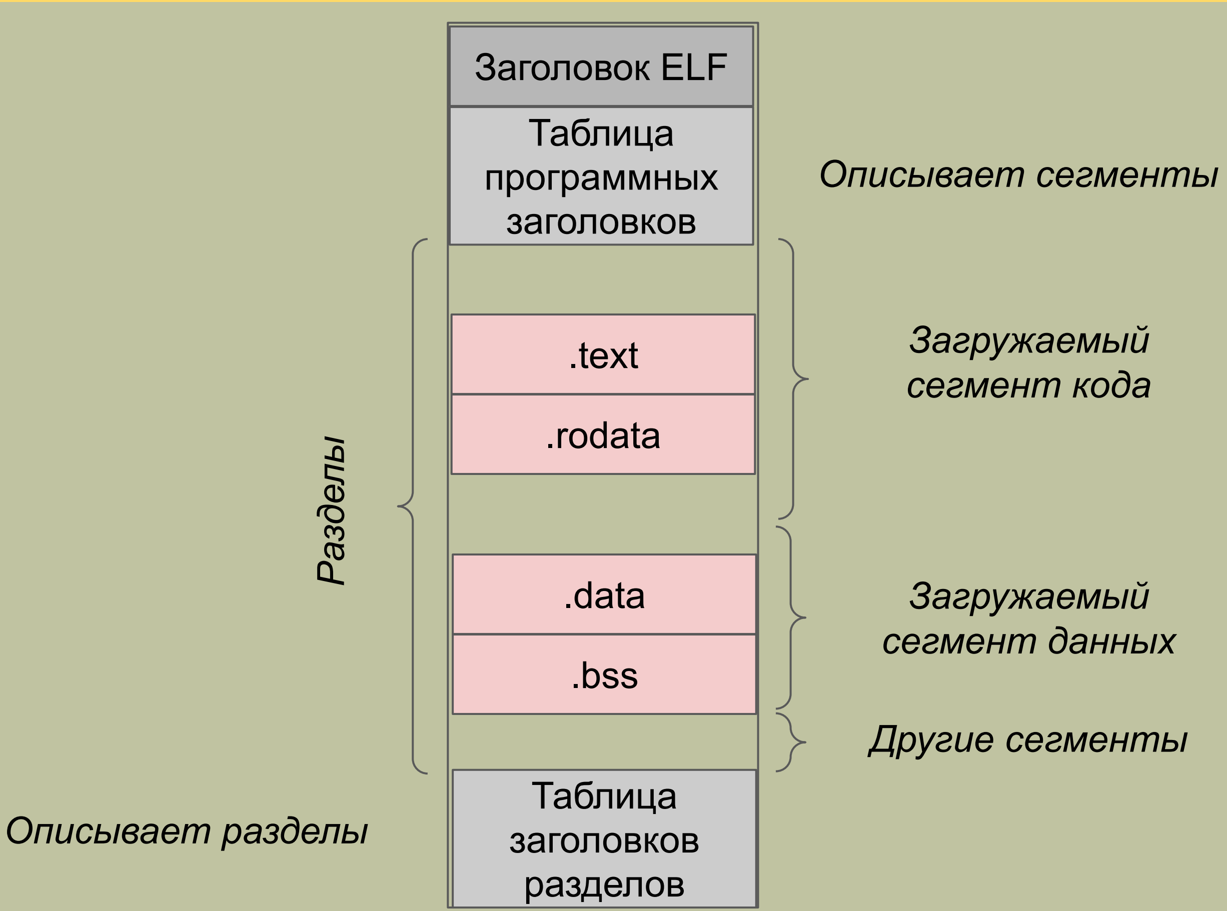
Лекция 6

- ELF файлы.

Структура ELF файла



```
Lab5> readelf -h liblab5.so
 Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00
00 00 00 00
 Класс:                              ELF64
........................................................................
 Тип:              DYN (Совм. исп. объектный файл)
 Машина:           Advanced Micro Devices X86-64
........................................................................
 Начало заголовков программы:64 (байт в  файле)
 Size of this header:                64 (bytes)
 Size of program headers:            56 (bytes)
 Number of program headers:          7
 ......................................................................
```

```
Lab5> readelf -l liblab5.so
```

```
Заголовки программы:
 Тип            Смещ.              Вирт.адр           Физ.адр
                Рзм.фйл            Рзм.пм              Флаги  Выравн
 LOAD           0x0000000000000000 0x0000000000000000 0x0000000000000000
                0x000000000000078c 0x000000000000078c  R E    0x200000
 LOAD           0x0000000000000e30 0x0000000000200e30 0x0000000000200e30
                0x0000000000000200 0x0000000000000208  RW     0x200000
 DYNAMIC        0x0000000000000e40 0x0000000000200e40 0x0000000000200e40
                0x00000000000001a0 0x00000000000001a0  RW     0x8
 NOTE           0x00000000000001c8 0x00000000000001c8 0x00000000000001c8
                0x0000000000000024 0x0000000000000024  R      0x4
 GNU_EH_FRAME   0x0000000000000698 0x0000000000000698 0x0000000000000698
                0x0000000000000034 0x0000000000000034  R      0x4
 GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                0x0000000000000000 0x0000000000000000  RW     0x10
 GNU_RELRO      0x0000000000000e30 0x0000000000200e30 0x0000000000200e30
                0x00000000000001d0 0x00000000000001d0  R      0x1
```

~Lab5> dumpelf liblab5.so

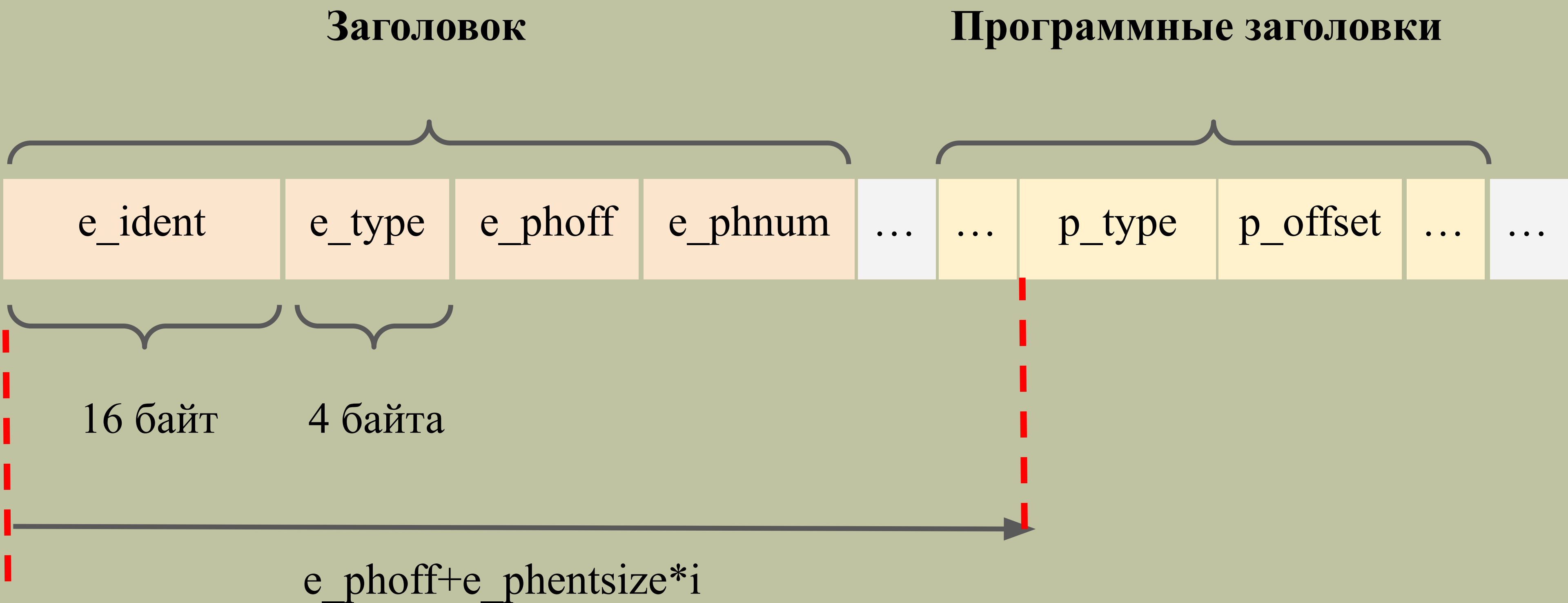
```
.phdrs = {
/* Program Header #0 0x40 */
{
       .p_type   = 1          , /* [PT_LOAD] */
       .p_offset = 0          , /* (bytes into file) */
       .p_vaddr  = 0x0        , /* (virtual addr at runtime)
*/
       .p_paddr  = 0x0        , /* (physical addr at runtime)
*/
       .p_filesz = 1932       , /* (bytes in file) */
       .p_memsz  = 1932       , /* (bytes in mem at runtime)
*/
       .p_flags  = 0x5        , /* PF_R | PF_X */
       .p_align  = 2097152    , /* (min mem alignment in
bytes) */
},
```

Заголовок ELF файла

```
typedef struct {
        unsigned char e_ident[EI_NIDENT];
        uint16_t     e_type;
        uint16_t     e_machine;
        uint32_t     e_version;
        ElfN_Addr      e_entry;
        ElfN_Off      e_phoff;
        ElfN_Off      e_shoff;
        uint32_t     e_flags;
        uint16_t     e_ehsize;
        uint16_t     e_phentsize;
        uint16_t     e_phnum;
        uint16_t     e_shentsize;
        uint16_t     e_shnum;
        uint16_t     e_shstrndx;
      } ElfN_Ehdr;
```

Таблица программных заголовков

```
typedef struct {
        uint32_t   p_type;
        uint32_t   p_flags;
        Elf64_Off  p_offset;
        Elf64_Addr p_vaddr;
        Elf64_Addr p_paddr;
        uint64_t   p_filesz;
        uint64_t   p_memsz;
        uint64_t   p_align;
      } Elf64_Phdr;
```



Заголовок

Программные заголовки

e_ident

e_type

e_phoff

e_phnum

…

…

p_type

p_offset

…

…

16 байт

4 байта

e_phoff+e_phentsize*i

```c
#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_Phdr  phheader;
 int i;
 FILE* file = fopen(elfFile, "rb");
```

```
fread(&header, sizeof(header), 1, file);
fclose(file);

for(i=0;i<16;i++)
  fprintf(stdout, "%x\t", header.e_ident[i]);
fprintf(stdout, "\n");

fprintf(stdout, "type: %x\t machine: %x\n",
            header.e_type, header.e_machine);
fprintf(stdout, "e_phoff: %x\n",
                             header.e_phoff);
fprintf(stdout, "e_phnum: %d\n",
                             header.e_phnum);
```

```
file = fopen(elfFile, "rb");

fseek(file,header.e_phoff, SEEK_SET);
 for(i=0;i<header.e_phnum;i++){
  if(i>0)
   fseek(file,
          header.e_phoff+header.e_phentsize*i,
          SEEK_SET);
  fread(&phheader, header.e_phentsize, 1,
        file);
  fprintf(stdout, "%x\t%x\t%x\t%x\n",
          phheader.p_type, phheader.p_offset,
          phheader.p_vaddr, phheader.p_paddr);
 fprintf(stdout, "%x\t%x\t%x\t%x\n",
         phheader.p_filesz, phheader.p_memsz,
        phheader.p_flags, phheader.p_align);
 fprintf(stdout, "\n");
}

 fclose(file);
 return 0;
}
```

```
/Lab5> ./lab5-elf liblab5.so
7f      45      4c      46      2       1
1       0       0       0       0       0
0  0
0       0
type: 3  machine: 3e
e_phoff: 40
e_phnum: 7
1       0       0       0
78c     78c     5       200000

1       e30     200e30  200e30
200     208     6       200000
```

```
2       e40     200e40  200e40
1a0     1a0     6       8

4       1c8     1c8     1c8
24      24      4       4

6474e550        698     698     698
34      34      4       4

6474e551        0       0       0
0       0       6       10
6474e552        e30     200e30  200e30
1d0     1d0     4       1
```

p_flags This member holds a bit mask of flags relevant to the segment:

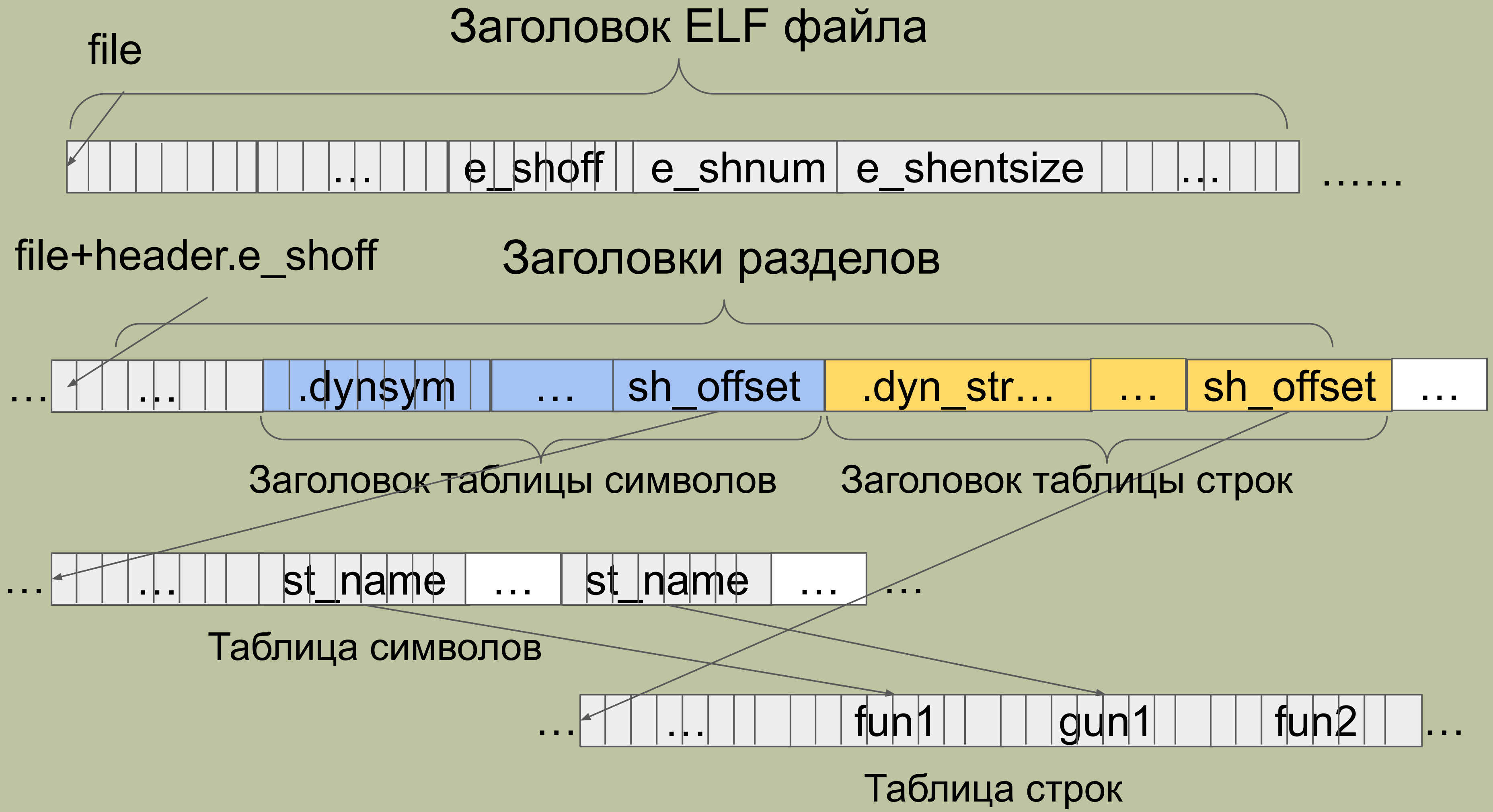
PF_X	An executable segment.
PF_W	A writable segment.
PF_R	A readable segment.

Таблица заголовков разделов

```
typedef struct {
        uint32_t   sh_name;
        uint32_t   sh_type;
        uint64_t   sh_flags;
        Elf64_Addr sh_addr;
        Elf64_Off  sh_offset;
        uint64_t   sh_size;
        uint32_t   sh_link;
        uint32_t   sh_info;
        uint64_t   sh_addralign;
        uint64_t   sh_entsize;
      } Elf64_Shdr;
```

Таблица символов

```
typedef struct {
        uint32_t      st_name;
        unsigned char st_info;
        unsigned char st_other;
        uint16_t      st_shndx;
        Elf64_Addr    st_value;
        uint64_t      st_size;
      } Elf64_Sym;
```



Заголовок ELF файла

file

… e_shoff e_shnum e_shentsize … ……

file+header.e_shoff

Заголовки разделов

… … .dynsym … sh_offset .dyn_str… … sh_offset …

Заголовок таблицы символов

Заголовок таблицы строк

… … st_name … st_name … …

Таблица символов

… … fun1 gun1 fun2 …

Таблица строк

lab5-elf5.c

```
#include <elf.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char** argv){
  //const char* elfFile="liblab5.so";
  const char*  elfFile=argv[1];
  Elf64_Ehdr  header;
  Elf64_Shdr  sheader;
  Elf64_Shdr  symtab;
  Elf64_Shdr  strtab;
  Elf64_Sym sym;
  char sname[32];
  int i;
  FILE* file = fopen(elfFile, "rb");
```

```
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;
}
```

```
   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;
}
```

```
/Lab5> ./lab5-elf5 liblab5.so
0     0     0     0     0
1     0     0     2     0     _ITM_deregisterTMCloneTable
2     0     0     2     0     __gmon_start__
3     0     0     2     0     _ITM_registerTMCloneTable
4     0     2     2     0     __cxa_finalize
5     24    2     1     12    fun1
6     56    2     1     12    gun1
7     22    2     1     12    fun2
8     8     1     1     22    y
9     0     2     1     9     _init
10    8     1     1     22    z
11    0     2     1     13    _fini
```

```
/Lab5> readelf -s liblab5.so

Symbol table '.dynsym' contains 12 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterT[...]
     2: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     3: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_registerTMC[...]
     4: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND [...]@GLIBC_2.2.5 (2)
     5: 000000000000061a    24 FUNC    GLOBAL DEFAULT   12 fun1
     6: 0000000000000648    56 FUNC    GLOBAL DEFAULT   12 gun1
     7: 0000000000000632    22 FUNC    GLOBAL DEFAULT   12 fun2
     8: 0000000000201020     8 OBJECT  GLOBAL DEFAULT   22 y
     9: 0000000000000508     0 FUNC    GLOBAL DEFAULT    9 _init
    10: 0000000000201028     8 OBJECT  GLOBAL DEFAULT   22 z
    11: 0000000000000680     0 FUNC    GLOBAL DEFAULT   13 _fini
```