ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Сибирский государственный университет
телекоммуникаций и информатики»
(СибГУТИ)

Отчет по лабораторной работе №3
по дисциплине “Системное программное обеспечение”

Выполнил:
студент гр. ИС-441
Брагин А.С.
Проверил:
Мамойленко С. Н.

Подпись_____

Дата_____

Новосибирск, 2017

Задание на лабораторную работу

1. Напишите программу, которая динамически выделяет 100 блоков памяти по 1000 байт каж- дый и затем освобождает их. Продемонстрируйте работоспособность Вашей программы.

2. Напишите динамическую библиотеку, в которой реализуйте две функции: `malloc` и `free`. Функция `malloc` выделяет (с использованием стандартной функции `malloc` из библиоте- ки `GLIBC`) запрашиваемый блок памяти до тех пор, пока не будет выделено 77 блоков. Далее любой вызов функции `malloc` приводит к ошибке выделения памяти. Функция `free` печа- тает на экране сообщение о количестве выделенных блоков памяти и освобождает запраши- ваемый блок).

3. Используя принудительную загрузку библиотек продемонстрируйте работу созданной в п.2 библиотеки на примере созданной в п.1 программы.

Выполнение работы

Программа, которая динамически выделяет 100 блоков памяти по 1000 байт каждый и затем освобождает их:

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

  #define INTERNAL_SIZE_T size_t
  #define SIZE_SZ (sizeof(INTERNAL_SIZE_T))
  #define MALLOC_ALIGN_MASK 2 * SIZE_SZ
  #define MIN_CHUNK_SIZE (offsetof(struct malloc_chunk, fd_nextsize))
  #define MINSIZE (unsigned long)
(((MIN_CHUNK_SIZE+MALLOC_ALIGN_MASK)&~MALLOC_ALIGN_MASK))
  #define chunk2mem(p) ((void*)((char*)(p) + 2*SIZE_SZ))
  #define mem2chunk(mem) ((mchunkptr)((char*)(mem) - 2*SIZE_SZ))

  struct malloc_chunk
  {
      INTERNAL_SIZE_T prev_size;
      INTERNAL_SIZE_T size;
      struct malloc_chunk* fd;
      struct malloc_chunk* bk;
      struct malloc_chunk* fd_nextsize;
      struct malloc_chunk* bk_nextsize;
  };

  typedef struct malloc_chunk* mchunkptr;

  #define request2size(req) (((req) + SIZE_SZ + MALLOC_ALIGN_MASK < MINSIZE) ?
MINSIZE : ((req) + SIZE_SZ + MALLOC_ALIGN_MASK) & ~MALLOC_ALIGN_MASK)

  int main()
  {
      int count = 100;
      int size = 1000;
      int *p[count];

      for (int i = 0; i < count; i++) {
          p[i] = malloc(request2size(i*size));
          printf ("iter = %d\n", i+1);
      }

     printf("\n\n");

     for(int i = 0; i < count; i++) free(p[i]);

      //malloc_stats();

      return 0;
  }
```

Динамическая библиотека:

```
#include <stdio.h>
#include <stdlib.h>
#define __USE_GNU
#define _GNU_SOURCE
#include <dlfcn.h>
#include <stdint.h>

static int iter = 0;

void *malloc (size_t size) {
    static void *(*real_malloc) (size_t) = NULL;

    if (iter >= 77) {
         fprintf (stderr, "\n!!! >77 !!!\n");
         return (NULL);
    }

    iter++;

    real_malloc = dlsym(RTLD_NEXT, "malloc");
    return real_malloc(size);
}

void free (void *ptr) {
   static void (*real_free) (void*) = NULL;
   if (iter - 1 < -1) return;

    fprintf (stderr, "Blocks: %d\n", iter);
   iter--;
    real_free = dlsym(RTLD_NEXT, "free");
   real_free(ptr);
}
```

Компиляция:

```
anton@anton-VirtualBox:~/Documents/SisOS$ gcc -fPIC -shared -o malloc.so malloc.c -ldl
anton@anton-VirtualBox:~/Documents/SisOS$ gcc l2.c -o l2 -rdynamic
anton@anton-VirtualBox:~/Documents/SisOS$ LD_PRELOAD="./malloc.so" ./l2
```

Результаты:

```
iter = 60
iter = 61
iter = 62
iter = 63
iter = 64
iter = 65
iter = 66
iter = 67
iter = 68
iter = 69
iter = 70
iter = 71
iter = 72
iter = 73
iter = 74
iter = 75
iter = 76

!!! >77 !!!
iter = 77

!!! >77 !!!
iter = 78

!!! >77 !!!
iter = 79

!!! >77 !!!
iter = 80

!!! >77 !!!
iter = 81

!!! >77 !!!
iter = 82

!!! >77 !!!
iter = 83
```

```
!!! >77 !!!
iter = 95

!!! >77 !!!
iter = 96

!!! >77 !!!
iter = 97

!!! >77 !!!
iter = 98

!!! >77 !!!
iter = 99

!!! >77 !!!
iter = 100


Blocks: 77
Blocks: 76
Blocks: 75
Blocks: 74
Blocks: 73
Blocks: 72
Blocks: 71
Blocks: 70
Blocks: 69
Blocks: 68
Blocks: 67
Blocks: 66
Blocks: 65
Blocks: 64
Blocks: 63
Blocks: 62
Blocks: 61
Blocks: 60
```

```
Blocks: 30
Blocks: 29
Blocks: 28
Blocks: 27
Blocks: 26
Blocks: 25
Blocks: 24
Blocks: 23
Blocks: 22
Blocks: 21
Blocks: 20
Blocks: 19
Blocks: 18
Blocks: 17
Blocks: 16
Blocks: 15
Blocks: 14
Blocks: 13
Blocks: 12
Blocks: 11
Blocks: 10
Blocks: 9
Blocks: 8
Blocks: 7
Blocks: 6
Blocks: 5
Blocks: 4
Blocks: 3
Blocks: 2
Blocks: 1
Blocks: 0
anton@anton-VirtualBox:~/Documents/SisOS$
```