Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра №806 «Вычислительная математика и программирование»

Лабораторная работа №4 по курсу

«Операционные системы»

Группа: М8О-211Б-23

Студент: Бачурин Н.В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 31.12.2024

Москва, 2024

Постановка задачи

Требуется создать две динамические библиотеки, реализующие два аллокатора, соответственно. Библиотеки загружаются в память с помощью интерфейса ОС (dlopen / LoadLibrary) для работы с динамическими библиотеками. Выбор библиотеки, реализующей аллокатор, осуществляется чтением первого аргумента при запуске программы (argv[1]). Этот аргумент должен содержать путь до динамической библиотеки (относительный или абсолютный).

Если аргумент не передан или по переданному пути библиотеки не оказалось, то указатели на функции, реализующие API аллокатора ниже, должны быть присвоены функциям, которые оборачивают системный аллокатор ОС (mmap / VirtualAlloc) в этот API. Эти аварийные оберточные функции должны быть реализованы внутри программы, которая загружает динамические библиотеки (см. пример на GitHub Gist).

Каждый аллокатор памяти должен иметь функции аналогичные стандартным функциям malloc и free (realloc, опционально). Перед работой каждый аллокатор инициализируется свободными страницами памяти, выделенными стандартными средствами ядра (mmap / VirtualAlloc). Необходимо самостоятельно разработать стратегию тестирования для определения ключевых характеристик аллокаторов памяти. При тестировании нужно свести к минимуму потери точности из-за накладных расходов при измерении ключевых характеристик, описанных выше.

Вариант 3 - Списки свободных блоков (первое подходящее) и алгоритм двойников

Общий метод и алгоритм решения

Использованные системные вызовы:

- void *mmap (void *address, size_t length, int protect, int flags, int filedes, off_t offset); - функция для сопоставления адресного пространства процесса с файлами или устройствами.
- ssize_t write(int fd, const void * buf, size_t n); – Записывает N байт из буфер(BUF) в файл (FD). Возвращает количество записанных байт или -1.
- void exit(int status); – выполняет немедленное завершение программы. Все используемые программой потоки закрываются, и временные файлы удаляются, управление возвращается ОС или другой программе.
- void *dlopen(const char *filename, int flag); – загружает динамическую библиотеку, имя которой указано в строке filename, и возвращает прямой указатель на начало динамической библиотеки.
- int dlclose(void *handle); – уменьшает на единицу счетчик ссылок на указатель динамической библиотеки.
- const char *dlerror(void); - возвращает понятную человеку, строку с null в конце, описывающую последнюю ошибку, которая произошла при вызове одной из функций программного интерфейса dlopen
- void *dlsym(void *handle, char *symbol); – функция ищет значение символа в открытой динамически загружаемой библиотеке. Она использует указатель на динамическую библиотеку, возвращаемую dlopen(), и оканчивающееся нулём символьное имя, а затем возвращает адрес;

Программа реализует алгоритм организации аллокатора памяти.
Основной файл main.c

- Проверяет наличие аргумента командной строки, указывающего на библиотеку аллокатора.
- Загружает библиотеку с помощью dlopen и загружает указатели на функции аллокатора с помощью dlsym.
- Если библиотека не загружена или функции не найдены, программа использует системный аллокатор.
- Создаёт аллокатор, выделяет блок памяти, замеряет время выделения и освобождения памяти, и затем освобождает аллокатор и библиотеку.

На вход main.c через argv[1] поступает путь до библиотеки аллокатора

Первый вариант: Аллокатор памяти на основе списка свободных блоков (первое подходящее)

Аллокатор памяти на основе списка свободных блоков использует структуру данных например, связанный список, для управления блоками памяти. Когда требуется выделить память, аллокатор ищет подходящий свободный блок в списке. После использования блоки возвращаются обратно в список.

Основные компоненты и принципы работы:

Список свободных блоков представляет собой структуру данных (обычно односвязный или двусвязный список), где каждый элемент описывает свободный участок памяти.

Основные компоненты:

- Список свободных блоков — структура данных (обычно связанный список), хранящая указатели на свободные участки памяти.
- Блоки памяти — участки памяти, которые аллокатор выделяет или освобождает. Каждый блок может содержать информацию о размере блока и указатель на следующий свободный блок.
- Функции выделения и освобождения памяти:
  1. Выделение: поиск подходящего свободного блока в списке.
  2. Освобождение: возвращение блока в список свободных.

Операции:

1. Инициализация свободного списка:

- Аллокатор получает блок памяти размером, достаточным для управления списком.
- Первый блок в списке представляет собой весь доступный фрагмент памяти.
- Этот блок выделяется как единственный в списке и содержит указатель на следующий свободный блок.

2. Выделение памяти:

- При запросе памяти, аллокатор ищет подходящий свободный блок в списке (если размер блока в списке больше или равен запрашиваемому размеру этот блок выделяется).
- Если блок больше требуемого, то его можно разделить на два блока: один с нужным размером, второй с оставшимся.
- В случае выделения памяти, блок удаляется из списка свободных блоков, а указатель на выделенную память возвращается в программу.
- 3. Освобождение памяти:
- Если блок памяти не нужен, то он освобождается и переходит в список свободных блоков.
- Если несколько блоков в списке свободных блоков могут быть объединены в один больший блок памяти.

Преимущества:

- Простота реализации: основные алгоритмы достаточно просты для понимания и кодирования.
- Гибкость: может работать с любыми размерами блоков.
- Минимальные требования к памяти: аллокатор требует минимального дополнительного пространства. Он использует саму область памяти для хранения информации о свободных блоках, добавляя только указатель на следующий блок и его размер.
- Недостатки:
- Фрагментация: при выделении и удалении блоков в разное время, может образоваться множество блоков с размеров, не подходящим под задачи.
- Невысокая производительность: операции выделения и освобождения памяти не самые быстрые операции из-за необходимости перебора списка свободных блоков.
- Отсутствие быстрого поиска: список свободных блоков не поддерживает эффективных алгоритмов поиска, например, бинарного поиска.

Второй вариант: Аллокатор памяти на основе алгоритма двойников

Это способ динамического распределения памяти, который помогает эффективно управлять памятью, минимизируя фрагментацию и улучшая производительность при выделении и освобождении блоков памяти. Данный метод один из популярных методов аллокации памяти, который находит применение в операционных системах и системах с жесткими требованиями к производительности и эффективности.

Основные компоненты и принципы работы:

Метод использует массив или список для хранения блоков памяти по их размерам (показатели порядка или степени двойки). Каждый блок в списке представляет собой указатель на свободный

блок определенного размера, и в случае выделения или освобождения памяти эти списки обновляются.

Операции:

1) Выделение памяти:

- При выделении памяти под определенный размер, алгоритм ищет минимальный блок, который подходит под требования. Если такой блок не удается найти, то алгоритм будет увеличивать блок до ближайшей степени двойки, пока блок не станет доступным.
- Таким же методом он может разделять большие блоки на более мелкие.

2) Освобождение памяти:

- Алгоритм проверяет, свободен ли соседний блок памяти. Если так, то 2 блока объединяются в 1 больший блок, который становится доступным.

3) Поиск и объединение блоков:

- После того, как произошло освобождение памяти и 2 блока сливаются в один, процесс может рекурсивно повторяться, если соседние блоки так же свободны.

Преимущества:

- Эффективное использование памяти.
- Меньше фрагментации. Память, которая выделяется и освобождается, может быть эффективно объединена в более крупные блоки.
- Быстрота выделения памяти. Алгоритм позволяет быстро выделять блоки, поскольку размер каждого блока — это степень двойки.
- Гибкость. Этот метод работает хорошо с любыми размерами блоков.

Недостатки:

- Внутренняя фрагментация. Поскольку блоки всегда выделяются в степени двойки, если размер запрашиваемой памяти не является степенью двойки, алгоритм может выделить блок большего размера, чем требуется. Это может привести к неэффективному использованию памяти.
- Сложность с большими блоками. Может потребоваться достаточно много времени, если самый большой блок значительно меньше, чем нужный размер памяти.
- Возможное увеличение времени для освобождения. Алгоритм может требовать много времени на объединение блоков при освобождении.

Код программы

main.c

```c
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>
#include <time.h>
#include <sys/mman.h>
#include <unistd.h>

#include "first_list_allocator.h"
#include "twin_allocator.h"

#define MEMORY_SIZE (1 << MAX_TWIN_ORDER)
char global_memory[MEMORY_SIZE];

typedef struct Object {
    int id;
    char name[50];
    float value;
} Object;


typedef void* (*allocator_create_t)(void* memory, size_t size);
typedef void (*allocator_destroy_t)(void* allocator);
typedef void* (*allocator_alloc_t)(void* allocator, size_t size);
typedef void (*allocator_free_t)(void* allocator, void* memory);


void* system_allocator_create(void* memory, size_t size) {
     return memory;
}

void system_allocator_destroy(void* allocator) {

}

void* system_allocator_alloc(void* allocator, size_t size) {
    if (size == 0) return NULL;

    void* block = mmap(NULL, size, PROT_READ | PROT_WRITE, MAP_PRIVATE | MAP_ANONYMOUS, -
1, 0);
     if (block == MAP_FAILED) {
        return NULL;
    }
    return block;
}

void system_allocator_free(void* allocator, void* memory) {
    if (!memory) return;
    size_t page_size = sysconf(_SC_PAGE_SIZE);
     munmap(memory, page_size);
}
```

```c
int main(int argc, char *argv[]) {
    allocator_create_t allocator_create = NULL;
    allocator_destroy_t allocator_destroy = NULL;
    allocator_alloc_t allocator_alloc = NULL;
    allocator_free_t allocator_free = NULL;

    void* allocator_lib = NULL;
    char* lib_path = NULL;

    if (argc > 1) {
      lib_path = argv[1];
      allocator_lib = dlopen(lib_path, RTLD_LAZY);
        if (!allocator_lib) {
            fprintf(stderr, "Ошибка загрузки библиотеки %s: %s\n", lib_path, dlerror());
            lib_path = NULL;
        }
    }

    if (lib_path) {
        if (strstr(lib_path, "liballocator.so")) {
              allocator_create = dlsym(allocator_lib, "allocator_create");
            allocator_destroy = dlsym(allocator_lib, "allocator_destroy");
            allocator_alloc = dlsym(allocator_lib, "allocator_alloc");
            allocator_free = dlsym(allocator_lib, "allocator_free");

        } else if (strstr(lib_path, "libtwin_allocator.so")) {
            allocator_create = dlsym(allocator_lib, "twin_allocator_create");
            allocator_destroy = dlsym(allocator_lib, "twin_allocator_destroy");
            allocator_alloc = dlsym(allocator_lib, "twin_allocator_alloc");
            allocator_free = dlsym(allocator_lib, "twin_allocator_free");
        }
    }

    if (!allocator_create || !allocator_alloc || !allocator_free) {
         fprintf(stderr, "Ошибка: не удалось найти функции API аллокатора в библиотеке.
Используется системный аллокатор.\n");
        allocator_create = (allocator_create_t)system_allocator_create;
        allocator_destroy = (allocator_destroy_t)system_allocator_destroy;
        allocator_alloc = (allocator_alloc_t)system_allocator_alloc;
        allocator_free = (allocator_free_t)system_allocator_free;

    }


    printf("Тестирование аллокатора:\n");
    void* allocator = allocator_create(global_memory, MEMORY_SIZE);
    if (!allocator) {
        fprintf(stderr, "Не удалось создать аллокатор.\n");
        if (allocator_lib) dlclose(allocator_lib);
        return 1;
    }

    clock_t start, end;
```

```
    start = clock();
    void* block = allocator_alloc(allocator, 64);
    end = clock();
    double alloc_time = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Выделен блок: %p, время: %.9f секунд\n", block, alloc_time);

    start = clock();
    allocator_free(allocator, block);
    end = clock();
    double free_time = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Освобождён блок: %p, время: %.9f секунд\n", block, free_time);

    allocator_destroy(allocator);

     if (allocator_lib) dlclose(allocator_lib);
    return 0;
}
```

First_list_allocator.h

```
#ifndef ALLOCATOR_H
#define ALLOCATOR_H

#include <stddef.h>
#include <math.h>
typedef struct FreeBlock {
    size_t size;
    struct FreeBlock* next;
} FreeBlock;

typedef struct Allocator {
    void* memory_start;
    size_t memory_size;
    FreeBlock* free_list;
} Allocator;

Allocator* allocator_create(void* memory, size_t size);
void allocator_destroy(Allocator* allocator);
void* allocator_alloc(Allocator* allocator, size_t size);
void allocator_free(Allocator* allocator, void* memory);

#endif // ALLOCATOR_H
```

First_list_allocator.c

```
#include "first_list_allocator.h"
#include <math.h>
Allocator* allocator_create(void* memory, size_t size) {
    if (size < sizeof(FreeBlock)) return NULL;

    Allocator* allocator = (Allocator*)memory;
    allocator->memory_start = (char*)memory + sizeof(Allocator);
```

```
    allocator->memory_size = size - sizeof(Allocator);
    allocator->free_list = (FreeBlock*)allocator->memory_start;

    allocator->free_list->size = allocator->memory_size;
    allocator->free_list->next = NULL;

    return allocator;
}

void allocator_destroy(Allocator* allocator) {
    (void)allocator;
}

void* allocator_alloc(Allocator* allocator, size_t size) {
    if (size == 0) return NULL;

    FreeBlock* prev = NULL;
    FreeBlock* current = allocator->free_list;

    while (current) {
        if (current->size >= size + sizeof(FreeBlock)) {
            if (current->size > size + sizeof(FreeBlock)) {
                FreeBlock* new_block = (FreeBlock*)((char*)current + sizeof(FreeBlock) +
size);
                new_block->size = current->size - size - sizeof(FreeBlock);
                new_block->next = current->next;
                current->next = new_block;
            }

            if (prev) {
                prev->next = current->next;
            } else {
                allocator->free_list = current->next;
            }

            current->size = size;
            return (char*)current + sizeof(FreeBlock);
        }

        prev = current;
        current = current->next;
    }

    return NULL;
}

void allocator_free(Allocator* allocator, void* memory) {
    if (!memory) return;

    FreeBlock* block_to_free = (FreeBlock*)((char*)memory - sizeof(FreeBlock));
    block_to_free->next = allocator->free_list;
    allocator->free_list = block_to_free;
}
```

Twin_allocator.c

```c
#include "twin_allocator.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
TwinAllocator* twin_allocator_create(void* memory, size_t size) {
    if (size < (1 << MAX_TWIN_ORDER)) return NULL;

    TwinAllocator* allocator = (TwinAllocator*)memory;
    allocator->memory_start = (char*)memory + sizeof(TwinAllocator);
    allocator->memory_size = size - sizeof(TwinAllocator);

    for (int i = 0; i <= MAX_TWIN_ORDER; i++) {
        allocator->free_lists[i] = NULL;
    }

    size_t initial_order = (size_t)log2(size);
    allocator->free_lists[initial_order] = (FreeBlock*)allocator->memory_start;
    allocator->free_lists[initial_order]->size = size;
    allocator->free_lists[initial_order]->next = NULL;

    return allocator;
}

void* twin_allocator_alloc(TwinAllocator* allocator, size_t size) {
    if (size == 0) return NULL;

    size_t order = (size_t)ceil(log2(size + sizeof(FreeBlock)));

    for (size_t i = order; i <= MAX_TWIN_ORDER; i++) {
        if (allocator->free_lists[i]) {
            FreeBlock* block = allocator->free_lists[i];
            allocator->free_lists[i] = block->next;

            while (i > order) {
                i--;
                size_t block_size = 1 << i;
                FreeBlock* twin = (FreeBlock*)((char*)block + block_size);

                twin->size = block_size;
                twin->next = allocator->free_lists[i];
                allocator->free_lists[i] = twin;
            }


            block->size = (1 << order);
            void* return_ptr = (char*)block + sizeof(FreeBlock);
            *((FreeBlock**)((char*)return_ptr - sizeof(FreeBlock*))) = block;
            return return_ptr;
        }
    }

    return NULL;
```

```
}

void twin_allocator_free(TwinAllocator* allocator, void* memory) {
    if (!memory) return;

    // Получаем указатель на заголовок блока
    FreeBlock* block = *((FreeBlock**)((char*)memory - sizeof(FreeBlock*)));

    // Вычисляем порядок блока
    size_t order = (size_t)log2(block->size);

    while (order <= MAX_TWIN_ORDER) {

        size_t block_size = 1 << order;
        char* base_address = (char*)allocator->memory_start;
        size_t block_offset = (char*)block - base_address;
        size_t buddy_offset = block_offset ^ block_size;
        FreeBlock* buddy = (FreeBlock*)(base_address + buddy_offset);

        FreeBlock** current_list = &allocator->free_lists[order];
        FreeBlock* prev = NULL;
        FreeBlock* current = *current_list;

        while (current) {
            if (current == buddy) {
                if (prev) {
                    prev->next = current->next;
                } else {
                    *current_list = current->next;
                }
                break;
            }
            prev = current;
            current = current->next;
        }

        if (!current) {
            block->next = allocator->free_lists[order];
            allocator->free_lists[order] = block;
            return;
        }


        if (block > buddy) {
            FreeBlock* temp = block;
            block = buddy;
            buddy = temp;
        }
        block->size = block_size * 2;
        order++;
    }


    block->next = allocator->free_lists[order];
```

```
    allocator->free_lists[order] = block;
}


void twin_allocator_destroy(TwinAllocator* allocator) {

}
```

twin_allocator.h

```
#ifndef TWIN_ALLOCATOR_H
#define TWIN_ALLOCATOR_H

#include "first_list_allocator.h"
#include <stddef.h>
#include <math.h>
#define MAX_TWIN_ORDER 20

typedef struct TwinAllocator {
    void* memory_start;
    size_t memory_size;
    FreeBlock* free_lists[MAX_TWIN_ORDER + 1];
} TwinAllocator;

TwinAllocator* twin_allocator_create(void* memory, size_t size);
void twin_allocator_destroy(TwinAllocator* allocator);
void* twin_allocator_alloc(TwinAllocator* allocator, size_t size);
void twin_allocator_free(TwinAllocator* allocator, void* memory);
void twin_allocator_destroy(TwinAllocator* allocator);
#endif // TWIN_ALLOCATOR_H
```

Протокол работы программы

Тестирование:

```
lausniko@DESKTOP-MATHSNO:~/os/lab4.1$  ./memory_allocator libtwin_allocator.so
Тестирование аллокатора:
Выделен блок: 0x56391b2bc108, время: 0.000015000 секунд
Освобождён блок: 0x56391b2bc108, время: 0.000001000 секунд
lausniko@DESKTOP-MATHSNO:~/os/lab4.1$ ./memory_allocator liballocator.so
Тестирование аллокатора:
Выделен блок: 0x563593b2a068, время: 0.000002000 секунд
Освобождён блок: 0x563593b2a068, время: 0.000001000 секунд
lausniko@DESKTOP-MATHSNO:~/os/lab4.1$ ./memory_allocator
Ошибка: не удалось найти функции API аллокатора в библиотеке. Используется системный аллокатор.
Тестирование аллокатора:
Выделен блок: 0x7f0d51daa000, время: 0.000003000 секунд
Освобождён блок: 0x7f0d51daa000, время: 0.000005000 секунд
```

```
lausniko@DESKTOP-MATHSNO:~/os/lab4.1$  ./memory_allocator libtwin_allocator.so
Тестирование аллокатора:
Выделение нескольких блоков:
Блок 0: 0x5582d9fbc108, время: 0.000024000 секунд
Блок 1: 0x5582d9fbc188, время: 0.000001000 секунд
Блок 2: 0x5582d9fbc208, время: 0.000000000 секунд
Блок 3: 0x5582d9fbc288, время: 0.000000000 секунд
Блок 4: 0x5582d9fbc308, время: 0.000001000 секунд
Блок 5: 0x5582d9fbc388, время: 0.000001000 секунд
Блок 6: 0x5582d9fbc408, время: 0.000001000 секунд
Блок 7: 0x5582d9fbc488, время: 0.000001000 секунд
Блок 8: 0x5582d9fbc508, время: 0.000001000 секунд
Блок 9: 0x5582d9fbc588, время: 0.000001000 секунд

Освобождение нескольких блоков:
Освобождён блок 0: 0x5582d9fbc108, время: 0.000001000 секунд
Освобождён блок 1: 0x5582d9fbc188, время: 0.000001000 секунд
Освобождён блок 2: 0x5582d9fbc208, время: 0.000001000 секунд
Освобождён блок 3: 0x5582d9fbc288, время: 0.000001000 секунд
Освобождён блок 4: 0x5582d9fbc308, время: 0.000000000 секунд
Освобождён блок 5: 0x5582d9fbc388, время: 0.000001000 секунд
Освобождён блок 6: 0x5582d9fbc408, время: 0.000001000 секунд
Освобождён блок 7: 0x5582d9fbc488, время: 0.000001000 секунд
Освобождён блок 8: 0x5582d9fbc508, время: 0.000001000 секунд
Освобождён блок 9: 0x5582d9fbc588, время: 0.000000000 секунд
```

```
lausniko@DESKTOP-MATHSNO:~/os/lab4.1$  ./memory_allocator liballocator.so
Тестирование аллокатора:
Выделение нескольких блоков:
Блок 0: 0x558ace052068, время: 0.000002000 секунд
Блок 1: 0x558ace0520b8, время: 0.000001000 секунд
Блок 2: 0x558ace052108, время: 0.000001000 секунд
Блок 3: 0x558ace052158, время: 0.000000000 секунд
Блок 4: 0x558ace0521a8, время: 0.000001000 секунд
Блок 5: 0x558ace0521f8, время: 0.000001000 секунд
Блок 6: 0x558ace052248, время: 0.000000000 секунд
Блок 7: 0x558ace052298, время: 0.000001000 секунд
Блок 8: 0x558ace0522e8, время: 0.000000000 секунд
Блок 9: 0x558ace052338, время: 0.000001000 секунд

Освобождение нескольких блоков:
Освобождён блок 0: 0x558ace052068, время: 0.000000000 секунд
Освобождён блок 1: 0x558ace0520b8, время: 0.000001000 секунд
Освобождён блок 2: 0x558ace052108, время: 0.000001000 секунд
Освобождён блок 3: 0x558ace052158, время: 0.000001000 секунд
Освобождён блок 4: 0x558ace0521a8, время: 0.000001000 секунд
Освобождён блок 5: 0x558ace0521f8, время: 0.000000000 секунд
Освобождён блок 6: 0x558ace052248, время: 0.000001000 секунд
Освобождён блок 7: 0x558ace052298, время: 0.000001000 секунд
Освобождён блок 8: 0x558ace0522e8, время: 0.000000000 секунд
Освобождён блок 9: 0x558ace052338, время: 0.000001000 секунд
```

Strace

lausniko@DESKTOP-MATHSNO:~/os/lab4.1$ strace ./memory_allocator libtwin_allocator.so

execve("./memory_allocator", ["./memory_allocator", "libtwin_allocator.so"], 0x7ffe090b1578 /* 36 vars */) = 0

brk(NULL) = 0x56064d6f8000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd6f443eb0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fca5b1ee000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "tls/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

```
    openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD, "x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "./x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "./libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

    newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=37327, ...}, AT_EMPTY_PATH) = 0

    mmap(NULL, 37327, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fca5b1e4000

    close(3)                                = 0

    openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

    read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0"..., 832)
= 832
```

```
    pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784,
64) = 784

    pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48,
848) = 48

    pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"...,     68,
896) = 68

    newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

    pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784,
64) = 784

    mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fca5afbb000

    mprotect(0x7fca5afe3000, 2023424, PROT_NONE) = 0

    mmap(0x7fca5afe3000,                  1658880,                  PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fca5afe3000

    mmap(0x7fca5b178000,  360448,  PROT_READ,  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,  3,
0x1bd000) = 0x7fca5b178000

    mmap(0x7fca5b1d1000,                  24576,                  PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7fca5b1d1000

    mmap(0x7fca5b1d7000,                  52816,                  PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fca5b1d7000

    close(3)                                = 0

    mmap(NULL,  12288,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|MAP_ANONYMOUS,  -1,  0)  =
0x7fca5afb8000

    arch_prctl(ARCH_SET_FS, 0x7fca5afb8740) = 0

    set_tid_address(0x7fca5afb8a10)         = 81796

    set_robust_list(0x7fca5afb8a20, 24)     = 0

    rseq(0x7fca5afb90e0, 0x20, 0, 0x53053053) = 0

    mprotect(0x7fca5b1d1000, 16384, PROT_READ) = 0

    mprotect(0x56063c750000, 4096, PROT_READ) = 0

    mprotect(0x7fca5b228000, 8192, PROT_READ) = 0

    prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0

    munmap(0x7fca5b1e4000, 37327)           = 0

    openat(AT_FDCWD,                   "./glibc-hwcaps/x86-64-v3/libtwin_allocator.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

    openat(AT_FDCWD,                   "./glibc-hwcaps/x86-64-v2/libtwin_allocator.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
```

```
    openat(AT_FDCWD, "./tls/x86_64/x86_64/libtwin_allocator.so", O_RDONLY|O_CLOEXEC) =
-1 ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libtwin_allocator.so", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libtwin_allocator.so", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/libtwin_allocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./x86_64/x86_64/libtwin_allocator.so", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./x86_64/libtwin_allocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "./x86_64/libtwin_allocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "./libtwin_allocator.so", O_RDONLY|O_CLOEXEC) = 3

    read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) =
832

    getrandom("\xb4\xfb\xda\x9b\x57\x88\xf5\xfd", 8, GRND_NONBLOCK) = 8

    brk(NULL)                               = 0x56064d6f8000

    brk(0x56064d719000)                     = 0x56064d719000

    newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=15720, ...}, AT_EMPTY_PATH) = 0

    getcwd("/home/lausniko/os/lab4.1", 128) = 25

    mmap(NULL, 16432, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fca5b1e9000

    mmap(0x7fca5b1ea000,                  4096,                  PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7fca5b1ea000

    mmap(0x7fca5b1eb000,  4096,  PROT_READ,  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,  3,
0x2000) = 0x7fca5b1eb000

    mmap(0x7fca5b1ec000,                  8192,                  PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7fca5b1ec000

    close(3)                                = 0

    openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libm.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libm.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)
```

```
    openat(AT_FDCWD, "./tls/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./tls/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "./x86_64/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "./x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "./libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD,  "glibc-hwcaps/x86-64-v3/libm.so.6",  O_RDONLY|O_CLOEXEC)  =  -1
ENOENT (No such file or directory)

    openat(AT_FDCWD,  "glibc-hwcaps/x86-64-v2/libm.so.6",  O_RDONLY|O_CLOEXEC)  =  -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "tls/x86_64/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "tls/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "tls/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "tls/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD, "x86_64/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

    newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=37327, ...}, AT_EMPTY_PATH) = 0

    mmap(NULL, 37327, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fca5afae000

    close(3)                                = 0

    openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

    read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) =
832
```

```
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fca5aec7000

mmap(0x7fca5aed5000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7fca5aed5000

mmap(0x7fca5af51000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) = 0x7fca5af51000

mmap(0x7fca5afac000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x7fca5afac000

close(3)                                = 0

mprotect(0x7fca5afac000, 4096, PROT_READ) = 0

mprotect(0x7fca5b1ec000, 4096, PROT_READ) = 0

munmap(0x7fca5afae000, 37327)           = 0

newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}, AT_EMPTY_PATH) = 0

write(1, "\320\242\320\265\321\201\321\202\320\270\321\200\320\276\320\262\320\260\320\275\320\270\320\265 \320\260\320\273\320\273\320"..., 47Тестирование аллокатора:

) = 47

clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=8641900}) = 0

clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=8710600}) = 0

write(1, "\320\222\321\213\320\264\320\265\320\273\320\265\320\275\320\261\320\273\320\276\320\272: 0x56063"..., 78Выделен блок: 0x7fca5b1e9000, время: 0.000069000 секунд

) = 78

clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=8842800}) = 0

clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=8887200}) = 0

write(1, "\320\236\321\201\320\262\320\276\320\261\320\276\320\266\320\264\321\221\320\275\320\261\320\273\320\276\320\272: 0"..., 84Освобождён блок: 0x7fca5b1e9000, время: 0.000045000 секунд

) = 84

munmap(0x7fca5b1e9000, 16432)           = 0

munmap(0x7fca5aec7000, 942344)          = 0

exit_group(0)                           = ?

+++ exited with 0 +++
```

```
lausniko@DESKTOP-MATHSNO:~/os/lab4.1$ strace ./memory_allocator liballocator.so
```

```
    execve("./memory_allocator",     ["./memory_allocator",     "liballocator.so"],
0x7ffd069d7ef8 /* 36 vars */) = 0

    brk(NULL)                                  = 0x559bf0690000

    arch_prctl(0x3001 /* ARCH_??? */, 0x7ffeb1d8a650) = -1 EINVAL (Invalid argument)

    mmap(NULL,  8192,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|MAP_ANONYMOUS,  -1,  0)  =
0x7f6cf03c5000

    access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)

    openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD,  "./tls/x86_64/x86_64/libc.so.6",  O_RDONLY|O_CLOEXEC)  =  -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD, "glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "tls/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)
```

```
    openat(AT_FDCWD, "tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file
or directory)

    openat(AT_FDCWD, "x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)

    openat(AT_FDCWD,  "./tls/x86_64/x86_64/libc.so.6",  O_RDONLY|O_CLOEXEC)  =  -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)

    openat(AT_FDCWD, "./libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)

    openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

    newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=37327, ...}, AT_EMPTY_PATH) =
0

    mmap(NULL, 37327, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f6cf03bb000

    close(3)                                = 0

    openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

    read(3,   "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"...,
832) = 832
```

```
    pread64(3,  "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

    pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

    pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68

    newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

    pread64(3,  "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

    mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6cf0192000

    mprotect(0x7f6cf01ba000, 2023424, PROT_NONE) = 0

    mmap(0x7f6cf01ba000,               1658880,               PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f6cf01ba000

    mmap(0x7f6cf034f000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f6cf034f000

    mmap(0x7f6cf03a8000,                24576,                PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f6cf03a8000

    mmap(0x7f6cf03ae000,                52816,                PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f6cf03ae000

    close(3)                                = 0

    mmap(NULL,  12288,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|MAP_ANONYMOUS,  -1,  0)  = 0x7f6cf018f000

    arch_prctl(ARCH_SET_FS, 0x7f6cf018f740) = 0

    set_tid_address(0x7f6cf018fa10)         = 82413

    set_robust_list(0x7f6cf018fa20, 24)     = 0

    rseq(0x7f6cf01900e0, 0x20, 0, 0x53053053) = 0

    mprotect(0x7f6cf03a8000, 16384, PROT_READ) = 0

    mprotect(0x559bd4fff000, 4096, PROT_READ) = 0

    mprotect(0x7f6cf03ff000, 8192, PROT_READ) = 0

    prlimit64(0,          RLIMIT_STACK,          NULL,          {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

    munmap(0x7f6cf03bb000, 37327)           = 0

    openat(AT_FDCWD,                    "./glibc-hwcaps/x86-64-v3/liballocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
```

```
    openat(AT_FDCWD,                    "./glibc-hwcaps/x86-64-v2/liballocator.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/x86_64/liballocator.so", O_RDONLY|O_CLOEXEC) = -
1 ENOENT (No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/liballocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "./tls/x86_64/liballocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

    openat(AT_FDCWD, "./tls/liballocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD,  "./x86_64/x86_64/liballocator.so",  O_RDONLY|O_CLOEXEC)  =  -1
ENOENT (No such file or directory)

    openat(AT_FDCWD, "./x86_64/liballocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./x86_64/liballocator.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)

    openat(AT_FDCWD, "./liballocator.so", O_RDONLY|O_CLOEXEC) = 3

    read(3,    "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"...,
832) = 832

    getrandom("\x7b\x15\x0b\x05\x53\xd9\xd2\x3c", 8, GRND_NONBLOCK) = 8

    brk(NULL)                                  = 0x559bf0690000

    brk(0x559bf06b1000)                        = 0x559bf06b1000

    newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=15280, ...}, AT_EMPTY_PATH) =
0

    getcwd("/home/lausniko/os/lab4.1", 128) = 25

    mmap(NULL, 16424, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6cf03c0000

    mmap(0x7f6cf03c1000,                4096,                PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f6cf03c1000

    mmap(0x7f6cf03c2000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x7f6cf03c2000

    mmap(0x7f6cf03c3000,                8192,                PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f6cf03c3000

    close(3)                                   = 0

    mprotect(0x7f6cf03c3000, 4096, PROT_READ) = 0

    newfstatat(1,  "",  {st_mode=S_IFCHR|0620,  st_rdev=makedev(0x88,  0x5),  ...},
AT_EMPTY_PATH) = 0
```

```
    write(1,
"\320\242\320\265\321\201\321\202\320\270\321\200\320\276\320\262\320\260\320\275\3
20\270\320\265 \320\260\320\273\320\273\320"..., 47Тестирование аллокатора:

    ) = 47

    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=6322500}) = 0

    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=6418200}) = 0

    write(1,              "\320\222\321\213\320\264\320\265\320\273\320\265\320\275
\320\261\320\273\320\276\320\272:  0x559bd"...,  78Выделен  блок:  0x7f6cf03c0000,
время: 0.000096000 секунд

    ) = 78

    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=6564200}) = 0

    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=6619400}) = 0

    write(1,
"\320\236\321\201\320\262\320\276\320\261\320\276\320\266\320\264\321\221\320\275
\320\261\320\273\320\276\320\272: 0"..., 84Освобождён блок: 0x7f6cf03c0000, время:
0.000055000 секунд

    ) = 84

    munmap(0x7f6cf03c0000, 16424)           = 0

    exit_group(0)                           = ?

    +++ exited with 0 +++
```

Вывод

В ходе написания данной лабораторной работы я узнал об устройстве аллокаторов. Научился создавать, подключать и использовать динамические библиотеки. Были реализованы два алгоритма аллокации памяти, работающие через один API и подключаемые через динамические библиотеки. В результате тестирования данных двух алгоритмов можно сказать, что оба алгоритма относительно эффективно справляются с задачей выделения и освобождения памяти, различие между ними можно заметить лишь при выделении первого блока: в данном случае аллокатор, реализованный на списке свободных блоков справляется быстрее. Вероятно это связано с тем, что выделяемый блок разделялся алгоритмом двойников до нужного размера, засчёт чего произошла задержка по времени.