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-210Б-23

Студент: Гула Дмитрий Александрович

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 01.11.24

Москва, 2024

Постановка задачи

Цель работы:

Приобретение практических навыков в:

1. Создании аллокаторов памяти и их анализу;

2. Создании динамических библиотек и программ, использующие динамические библиотеки.

Задание

Исследовать два аллокатора памяти: необходимо реализовать два алгоритма аллокации памяти и сравнить их по следующим характеристикам: – Фактор использования – Скорость выделения блоков – Скорость освобождения блоков – Простота использования аллокатора 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 7. Списки свободных блоков (наиболее подходящее) и блоки по 2^n

Общий метод и алгоритм решения

Использованные системные вызовы:

write — вывод данных в стандартный поток вывода и ошибок.
malloc — выделение памяти в куче.
free — освобождение памяти, выделенной с помощью malloc.
dlopen — динамическая загрузка разделяемой библиотеки.
dlsym — получение адреса функции из загруженной библиотеки.
dlclose — закрытие загруженной библиотеки.
snprintf — форматирование строк.
strlen — определение длины строки.

Аллокатор памяти на основе списка свободных блоков

Это один из классических методов управления динамической памятью. Он основан на поддержании списка, в котором хранятся все доступные (свободные) участки памяти. Когда программе требуется выделить память, аллокатор ищет в этом списке подходящий свободный блок. После использования выделенную память возвращают обратно, и она снова попадает в список свободных блоков.

Основные компоненты и принципы работы:

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

● Размер блока: количество байтов, доступных в этом свободном блоке.
● Указатель на следующий свободный блок: используется для связывания элементов списка.
● Возможны дополнительные поля, такие как флаг “свободен” или метаинформация для целей отладки. Сама структура аллокатора обычно включает:
● Указатель на память;
● Указатель на голову списка;
● Размер памяти.

Преимущества:

● Простота реализации: основные алгоритмы достаточно просты для понимания и кодирования.
● Гибкость: может быть адаптирован к различным сценариям использования и требованиям.

Недостатки:
● Фрагментация: при частом выделении и освобождении блоков возникает фрагментация – множество мелких свободных блоков, непригодных для выделения под большие запросы.
● Зависимость от стратегии: эффективность сильно зависит от выбранной стратегии поиска свободного блока и сценария использования.
● Затраты на поиск: стратегия best-fit требует просмотра всего списка, что может быть затратно по времени при большом количестве свободных блоков блоков.

Аллокатор памяти на основе блоков 2^n

Этот тип аллокатора использует подход, основанный на выделении блоков памяти размером, являющимся степенью двойки. Он эффективно управляет выделением и освобождением памяти, минимизируя фрагментацию.

Основные компоненты и принципы работы:

Информация о свободных блоках хранится в массиве (m) списков свободных блоков, где индекс массива (i) удовлетворяет условию: m[i].size == min_block_size * 2^i Каждый элемент (блок) обычно содержит:
● Размер блока: количество байтов, доступных в этом свободном блоке (степень 2).
● Указатель на следующий свободный блок такого же размера: используется для связывания элементов списка.
● Возможны дополнительные поля, такие как флаг “свободен” или метаинформация для целей отладки. Сама структура аллокатора обычно включает:
● Указатель на память;
● Размер памяти;
● Массив списков (динамический или статический);
● Количество списков (если массив динамический);
● Минимальный размер блока.

Преимущества:
● Высокая скорость поиска: поиск блока определённого размера значительно упрощен, т.к. блоки разбиты по степеням двойки.
● Простая реализация: аллокатор имеет относительно простую структуру, что упрощает его разработку и отладку.
● Отсутствие внешней фрагментации: поскольку размеры блоков фиксированы (степени двойки), не возникает внешней фрагментации.
● Минимизация накладных расходов: отсутствие перераспределения и объединения блоков уменьшает накладные расходы на управление памятью.

Недостатки:
● Неэффективность для нестандартных запросов: если размер запроса не является степенью двойки, то аллокатор будет выделять блок памяти, большего размера, чем требуется.
● Внутренняя сегментация: оптимальный размер блока может не совпадать с реальными потребностями программы, что создаёт пустоты в памяти, которые нельзя использовать, до освобождения всего блока.
● Отсутствие слияния свободных блоков: может привести к отсутствию блоков большого размера после длительного использования одного аллокатора

Тест 1: Базовое выделение и освобождение

Цель: Проверить, что аллокатор может успешно выделить и освободить память для небольших блоков.

Шаги:

- Выделяется два блока памяти: один размером 100 байт, другой — 200 байт.
- Проверяется, что оба блока были успешно выделены.
-
- Оба блока освобождаются.

Ожидаемый результат: Программа должна вывести сообщение об успешном выделении памяти и адреса выделенных блоков. После освобождения памяти не должно быть утечек.

Тест 2: Выделение и освобождение множества блоков

Цель: Проверить, что аллокатор может обрабатывать множество запросов на выделение и освобождение памяти.

Шаги:

- В цикле выделяется 100 блоков памяти по 128 байт каждый.
- Проверяется, что каждый блок был успешно выделен.
- Все выделенные блоки освобождаются.

Ожидаемый результат: Программа должна вывести сообщения об успешном выделении каждого блока. После освобождения всех блоков не должно быть утечек памяти.

Тест 3: Проверка на утечки памяти

Цель: Проверить, что аллокатор корректно освобождает всю выделенную память и не допускает утечек.

Шаги:

- Выделяется два блока памяти: один размером 300 байт, другой — 400 байт.
- Проверяется, что оба блока были успешно выделены.
- Оба блока освобождаются.
- Сравнивается объем выделенной и освобожденной памяти.

Ожидаемый результат: Программа должна вывести сообщение об отсутствии утечек памяти, если объем выделенной и освобожденной памяти совпадает.

Тест 4: Выделение памяти до исчерпания

Цель: Проверить, как аллокатор ведет себя при исчерпании доступной памяти.

Шаги:

- В цикле выделяются блоки памяти по 128 байт до тех пор, пока аллокатор не вернет nullptr.
- Записывается общий объем выделенной памяти.
- Все выделенные блоки освобождаются.

Ожидаемый результат: Программа должна вывести сообщение о том, что память исчерпана, и указать общий объем выделенной памяти. После освобождения всех блоков не должно быть утечек.

Тест 5: Повторное выделение освобожденной памяти

Цель: Проверить, что аллокатор может повторно использовать освобожденную память.

Шаги:

- Выделяется блок памяти размером 100 байт.
- Блок освобождается.
- Выделяется новый блок памяти размером 100 байт.
- Проверяется, что новый блок был успешно выделен.
- Новый блок освобождается.

Ожидаемый результат: Программа должна вывести сообщения об успешном выделении и повторном выделении памяти. После освобождения блока не должно быть утечек.

.

Код программы

main.cpp:

```cpp
#include <iostream>

#include <dlfcn.h>

#include <cstdlib>

#include <ctime>

#include <vector>

#include <chrono>

#include <unistd.h>

#include <cstring>



typedef void* (*CreateFunc)(void*, size_t);
```

```c
typedef void (*DestroyFunc)(void*);

typedef void* (*AllocFunc)(void*, size_t);

typedef void (*FreeFunc)(void*, void*);


void* sys_alloc(size_t size) {
    return malloc(size);
}


void sys_free(void* ptr) {
    free(ptr);
}


void write_output(const char* message) {
    write(1, message, strlen(message));
}


void write_error(const char* message) {
    write(2, message, strlen(message));
}


int main(int argc, char* argv[]) {
    if (argc < 2) {
        write_error("Usage: ");
        write_error(argv[0]);
        write_error(" <path_to_allocator_library>\n");
        return 1;
    }


    void* handle = dlopen(argv[1], RTLD_LAZY);
```

```
    if (!handle) {
        write_error("Error loading library: ");
        write_error(dlerror());
        write_error("\n");
        return 1;
    }

    CreateFunc create = (CreateFunc)dlsym(handle, "allocator_create");
    DestroyFunc destroy = (DestroyFunc)dlsym(handle, "allocator_destroy");
    AllocFunc alloc = (AllocFunc)dlsym(handle, "allocator_alloc");
    FreeFunc free_func = (FreeFunc)dlsym(handle, "allocator_free");

    if (!create || !destroy || !alloc || !free_func) {
        write_error("Error loading functions: ");
        write_error(dlerror());
        write_error("\n");
        dlclose(handle);
        return 1;
    }

    size_t allocator_size = 1024 * 1024; // 1 MB
    void* allocator_memory = malloc(allocator_size);
    void* allocator = create(allocator_memory, allocator_size);

    // Тест 1: Базовое выделение и освобождение
    void* ptr1 = alloc(allocator, 100);
    void* ptr2 = alloc(allocator, 200);

    if (ptr1 && ptr2) {
```

```
        write_output("Allocation successful!\n");
        char buffer[128];
        snprintf(buffer, sizeof(buffer), "ptr1: %p (100 bytes)\n", ptr1);
        write_output(buffer);
        snprintf(buffer, sizeof(buffer), "ptr2: %p (200 bytes)\n", ptr2);
        write_output(buffer);
    } else {
        write_error("Allocation failed!\n");
        if (!ptr1) write_error("Failed to allocate 100 bytes\n");
        if (!ptr2) write_error("Failed to allocate 200 bytes\n");
    }

    free_func(allocator, ptr1);
    free_func(allocator, ptr2);

    write_output("\n");

    // Тест 2: Выделение и освобождение множества блоков
    std::vector<void*> pointers;
    for (int i = 0; i < 100; ++i) {
        void* ptr = alloc(allocator, 128);
        if (ptr) {
            pointers.push_back(ptr);
            char buffer[128];
            snprintf(buffer, sizeof(buffer), "%d ptr%p\n", i + 1, ptr);
            write_output(buffer);
        } else {
            char buffer[128];
            snprintf(buffer, sizeof(buffer), "Failed to allocate block %d\n",
i);
```

```
            write_error(buffer);
            break;
        }
    }

    for (void* ptr : pointers) {
        free_func(allocator, ptr);
    }

    write_output("\n");

    // Тест 3: Проверка на утечки памяти
    std::vector<void*> pointers_test3;
    size_t allocated_memory = 0;
    size_t freed_memory = 0;

    void* ptr3 = alloc(allocator, 300);
    if (ptr3) {
        pointers_test3.push_back(ptr3);
        allocated_memory += 300;
    }

    void* ptr4 = alloc(allocator, 400);
    if (ptr4) {
        pointers_test3.push_back(ptr4);
        allocated_memory += 400;
    }

    for (void* ptr : pointers_test3) {
```

```
        free_func(allocator, ptr);
        freed_memory += (ptr == ptr3) ? 300 : 400;
    }

    if (allocated_memory == freed_memory) {
        write_output("No memory leaks detected!\n");
    } else {
        write_error("Memory leak detected!\n");
    }

    write_output("\n");

    // Тест 4:  на выделение памяти до исчерпания

    std::vector<void*> pointers4;
    size_t total_allocated = 0;
    while (true) {
        void* ptr = alloc(allocator, 128);
        if (ptr) {
            pointers4.push_back(ptr);
            total_allocated += 128;
        } else {
            char buffer[128];
            snprintf(buffer, sizeof(buffer), "Memory exhausted after
allocating %zu bytes\n", total_allocated);
            write_output(buffer);
            break;
        }
    }
```

```
    for (void* ptr : pointers4) {
        free_func(allocator, ptr);
    }

    // Тест 5:  Тест на повторное выделение освобожденной памяти

    void* ptr10 = alloc(allocator, 100);
    if (ptr10) {
        char buffer[128];
        snprintf(buffer, sizeof(buffer), "Allocated 100 bytes at %p\n",
ptr10);
        write_output(buffer);
        free_func(allocator, ptr10);

        void* ptr20 = alloc(allocator, 100);
        if (ptr20) {
            snprintf(buffer, sizeof(buffer), "Reallocated 100 bytes at %p\n",
ptr20);
            write_output(buffer);
            free_func(allocator, ptr20);
        } else {
            write_error("Reallocation failed!\n");
        }
    } else {
        write_error("Initial allocation failed!\n");
    }

    destroy(allocator);
    free(allocator_memory);
    dlclose(handle);
```

```
    return 0;
}
```

allocator1.cpp:

```
#include <iostream>
#include <cstdlib>
#include <cstddef>

struct Block {
    size_t size;
    bool is_free;
    Block *next;
};

class Allocator {
private:
    void *memory;
    size_t size;
    Block *free_list;

public:
    Allocator(void *mem, size_t sz) {
        memory = (void *)((char *)mem + sizeof(Allocator));
        size = sz - sizeof(Allocator);
        free_list = (Block *)memory;
        free_list->size = size;
        free_list->is_free = true;
```

```
        free_list->next = nullptr;
    }

    ~Allocator() {}

    void *alloc(size_t size) {
        Block *current = free_list;
        Block *prev = nullptr;

        while (current) {
            if (current->is_free && current->size >= size) {
                if (current->size >= size + sizeof(Block)) {
                    Block *new_block = (Block *)((char *)current +
sizeof(Block) + size);
                    new_block->size = current->size - size - sizeof(Block);
                    new_block->is_free = true;
                    new_block->next = current->next;

                    current->size = size;
                    current->is_free = false;
                    current->next = new_block;
                }
                else {
                    current->is_free = false;
                }
                return (void *)((char *)current + sizeof(Block));
            }
            prev = current;
            current = current->next;
        }
```

```
        return nullptr;
    }

    void free(void* ptr) {
    if (!ptr) return;

    Block* block = (Block*)((char*)ptr - sizeof(Block));
    block->is_free = true;

    Block* current = free_list;
    Block* prev = nullptr;
    while (current && current != block) {
        prev = current;
        current = current->next;
    }

    if (prev && prev->is_free) {
        prev->size += sizeof(Block) + block->size;
        prev->next = block->next;
        block = prev;
    } else if (!current) {
        block->next = free_list;
        free_list = block;
    }

    if (block->next && block->next->is_free) {
        block->size += sizeof(Block) + block->next->size;
        block->next = block->next->next;
    }
```

```
}

};


extern "C" {

    Allocator *allocator_create(void *memory, size_t size) {

        return new (memory) Allocator(memory, size);

    }


    void allocator_destroy(Allocator *allocator) {

        allocator->~Allocator();

    }


    void *allocator_alloc(Allocator *allocator, size_t size) {

        return allocator->alloc(size);

    }


    void allocator_free(Allocator *allocator, void *ptr) {

        allocator->free(ptr);

    }

}
```

allocator2.cpp:

```
#include <iostream>

#include <cstdlib>

#include <cstddef>

#include <vector>


const size_t NUM_SIZES = 10;
```

```
const size_t BLOCK_SIZES[NUM_SIZES] = {16, 32, 64, 128, 256, 512, 1024,
2048, 4096, 8192};


class Allocator2 {
private:
    void* memory;
    size_t size;
    std::vector<void*> free_lists[NUM_SIZES];

public:
    Allocator2(void* mem, size_t sz) : memory(mem), size(sz) {
        for (size_t i = 0; i < NUM_SIZES; ++i) {
            free_lists[i].reserve(sz / BLOCK_SIZES[i]);
        }

        for (size_t i = 0; i < NUM_SIZES; ++i) {
            size_t block_size = BLOCK_SIZES[i];
            size_t num_blocks = sz / block_size;
            for (size_t j = 0; j < num_blocks; ++j) {
                void* block = (char*)memory + j * block_size;
                free_lists[i].push_back(block);
            }
        }
    }

    ~Allocator2() {
    }

    void* alloc(size_t size) {
        for (size_t i = 0; i < NUM_SIZES; ++i) {
```

```
            if (size <= BLOCK_SIZES[i]) {
                if (!free_lists[i].empty()) {
                    void* ptr = free_lists[i].back();
                    free_lists[i].pop_back();
                    return ptr;
                } else {
                    if ((char*)memory + BLOCK_SIZES[i] <= (char*)memory +
this->size) {
                        void* ptr = memory;
                        memory = (char*)memory + BLOCK_SIZES[i];
                        this->size -= BLOCK_SIZES[i];
                        return ptr;
                    }
                }
            }
        }
        return nullptr;
    }

    void free(void* ptr) {
        if (!ptr) return;

        for (size_t i = 0; i < NUM_SIZES; ++i) {
            if (BLOCK_SIZES[i] >= size) {
                free_lists[i].push_back(ptr);
                return;
            }
        }
    }
};
```

```
extern "C" {
    Allocator2* allocator_create(void* memory, size_t size) {
        return new (memory) Allocator2(memory, size);
    }

    void allocator_destroy(Allocator2* allocator) {
        allocator->~Allocator2();
    }

    void* allocator_alloc(Allocator2* allocator, size_t size) {
        return allocator->alloc(size);
    }

    void allocator_free(Allocator2* allocator, void* ptr) {
        allocator->free(ptr);
    }
}
```

Протокол работы программы

Тестирование:

```
dmitry@dmitry-HP-Laptop-15-da3xxx:~/C/Labs-OS/lab4/src$ ./main ./liballocator1.so
Allocation successful!
ptr1: 0x7cd20ea16040 (100 bytes)
ptr2: 0x7cd20ea160bc (200 bytes)

1 ptr0x7cd20ea16040
2 ptr0x7cd20ea160d8
3 ptr0x7cd20ea16170
4 ptr0x7cd20ea16208
5 ptr0x7cd20ea162a0
6 ptr0x7cd20ea16338
7 ptr0x7cd20ea163d0
8 ptr0x7cd20ea16468
```

9 ptr0x7cd20ea16500
10 ptr0x7cd20ea16598
11 ptr0x7cd20ea16630
12 ptr0x7cd20ea166c8
13 ptr0x7cd20ea16760
14 ptr0x7cd20ea167f8
15 ptr0x7cd20ea16890
16 ptr0x7cd20ea16928
17 ptr0x7cd20ea169c0
18 ptr0x7cd20ea16a58
19 ptr0x7cd20ea16af0
20 ptr0x7cd20ea16b88
21 ptr0x7cd20ea16c20
22 ptr0x7cd20ea16cb8
23 ptr0x7cd20ea16d50
24 ptr0x7cd20ea16de8
25 ptr0x7cd20ea16e80
26 ptr0x7cd20ea16f18
27 ptr0x7cd20ea16fb0
28 ptr0x7cd20ea17048
29 ptr0x7cd20ea170e0
30 ptr0x7cd20ea17178
31 ptr0x7cd20ea17210
32 ptr0x7cd20ea172a8
33 ptr0x7cd20ea17340
34 ptr0x7cd20ea173d8
35 ptr0x7cd20ea17470
36 ptr0x7cd20ea17508
37 ptr0x7cd20ea175a0
38 ptr0x7cd20ea17638
39 ptr0x7cd20ea176d0
40 ptr0x7cd20ea17768
41 ptr0x7cd20ea17800
42 ptr0x7cd20ea17898
43 ptr0x7cd20ea17930
44 ptr0x7cd20ea179c8
45 ptr0x7cd20ea17a60
46 ptr0x7cd20ea17af8
47 ptr0x7cd20ea17b90
48 ptr0x7cd20ea17c28
49 ptr0x7cd20ea17cc0
50 ptr0x7cd20ea17d58
51 ptr0x7cd20ea17df0
52 ptr0x7cd20ea17e88
53 ptr0x7cd20ea17f20
54 ptr0x7cd20ea17fb8
55 ptr0x7cd20ea18050
56 ptr0x7cd20ea180e8
57 ptr0x7cd20ea18180

```
58 ptr0x7cd20ea18218
59 ptr0x7cd20ea182b0
60 ptr0x7cd20ea18348
61 ptr0x7cd20ea183e0
62 ptr0x7cd20ea18478
63 ptr0x7cd20ea18510
64 ptr0x7cd20ea185a8
65 ptr0x7cd20ea18640
66 ptr0x7cd20ea186d8
67 ptr0x7cd20ea18770
68 ptr0x7cd20ea18808
69 ptr0x7cd20ea188a0
70 ptr0x7cd20ea18938
71 ptr0x7cd20ea189d0
72 ptr0x7cd20ea18a68
73 ptr0x7cd20ea18b00
74 ptr0x7cd20ea18b98
75 ptr0x7cd20ea18c30
76 ptr0x7cd20ea18cc8
77 ptr0x7cd20ea18d60
78 ptr0x7cd20ea18df8
79 ptr0x7cd20ea18e90
80 ptr0x7cd20ea18f28
81 ptr0x7cd20ea18fc0
82 ptr0x7cd20ea19058
83 ptr0x7cd20ea190f0
84 ptr0x7cd20ea19188
85 ptr0x7cd20ea19220
86 ptr0x7cd20ea192b8
87 ptr0x7cd20ea19350
88 ptr0x7cd20ea193e8
89 ptr0x7cd20ea19480
90 ptr0x7cd20ea19518
91 ptr0x7cd20ea195b0
92 ptr0x7cd20ea19648
93 ptr0x7cd20ea196e0
94 ptr0x7cd20ea19778
95 ptr0x7cd20ea19810
96 ptr0x7cd20ea198a8
97 ptr0x7cd20ea19940
98 ptr0x7cd20ea199d8
99 ptr0x7cd20ea19a70
100 ptr0x7cd20ea19b08

No memory leaks detected!

Memory exhausted after allocating 882944 bytes
Allocated 100 bytes at 0x7cd20ea16040
Reallocated 100 bytes at 0x7cd20ea16040
```

Dtruss:

```
dmitry@dmitry-HP-Laptop-15-da3xxx:~/C/Labs-OS/lab4/src$ strace ./main
./liballocator1.so
execve("./main", ["./main", "./liballocator1.so"], 0x7ffdc7bff4b8 /* 74 vars */) = 0
brk(NULL)                               = 0x5ea6f54d7000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x72d3d73d1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v4/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "./libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "glibc-hwcaps/x86-64-v4/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "glibc-hwcaps/x86-64-v3/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "glibc-hwcaps/x86-64-v2/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "libstdc++.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
fstat(3, {st_mode=S_IFREG|0644, st_size=58511, ...}) = 0
mmap(NULL, 58511, PROT_READ, MAP_PRIVATE, 3, 0) = 0x72d3d73c2000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=2592224, ...}) = 0
mmap(NULL, 2609472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x72d3d7000000
mmap(0x72d3d709d000, 1343488, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9d000) = 0x72d3d709d000
mmap(0x72d3d71e5000, 552960, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1e5000) = 0x72d3d71e5000
mmap(0x72d3d726c000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x26b000) = 0x72d3d726c000
mmap(0x72d3d727a000, 12608, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x72d3d727a000
close(3)                                = 0
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v4/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
```

```
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "./libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "glibc-hwcaps/x86-64-v4/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "glibc-hwcaps/x86-64-v3/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "glibc-hwcaps/x86-64-v2/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", 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"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=183024, ...}) = 0
mmap(NULL, 185256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x72d3d7394000
mmap(0x72d3d7398000, 147456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x4000) = 0x72d3d7398000
mmap(0x72d3d73bc000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000)
= 0x72d3d73bc000
mmap(0x72d3d73c0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2b000) = 0x72d3d73c0000
close(3)                                = 0
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v4/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, "./libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "glibc-hwcaps/x86-64-v4/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, "libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
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\0\220\243\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
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x72d3d6c00000
mmap(0x72d3d6c28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
```

```
3, 0x28000) = 0x72d3d6c28000
mmap(0x72d3d6db0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x72d3d6db0000
mmap(0x72d3d6dff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x72d3d6dff000
mmap(0x72d3d6e05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x72d3d6e05000
close(3)                                = 0
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v4/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, "./libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "glibc-hwcaps/x86-64-v4/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, "libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
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
fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0
mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x72d3d72ab000
mmap(0x72d3d72bb000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x72d3d72bb000
mmap(0x72d3d733a000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x72d3d733a000
mmap(0x72d3d7392000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x72d3d7392000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x72d3d72a9000
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x72d3d72a6000
arch_prctl(ARCH_SET_FS, 0x72d3d72a6740) = 0
set_tid_address(0x72d3d72a6a10)         = 64864
set_robust_list(0x72d3d72a6a20, 24)     = 0
rseq(0x72d3d72a7060, 0x20, 0, 0x53053053) = 0
mprotect(0x72d3d6dff000, 16384, PROT_READ) = 0
mprotect(0x72d3d7392000, 4096, PROT_READ) = 0
mprotect(0x72d3d73c0000, 4096, PROT_READ) = 0
mprotect(0x72d3d726c000, 45056, PROT_READ) = 0
mprotect(0x5ea6b7981000, 4096, PROT_READ) = 0
mprotect(0x72d3d7409000, 8192, PROT_READ) = 0
```

```
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x72d3d73c2000, 58511)           = 0
futex(0x72d3d727a7bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
getrandom("\x8c\x11\xaa\xc8\xd5\x1f\xa1\xd2", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5ea6f54d7000
brk(0x5ea6f54f8000)                     = 0x5ea6f54f8000
openat(AT_FDCWD, "./liballocator1.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
fstat(3, {st_mode=S_IFREG|0775, st_size=16256, ...}) = 0
getcwd("/home/dmitry/C/Labs-OS/lab4/src", 128) = 32
mmap(NULL, 16440, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x72d3d73cc000
mmap(0x72d3d73cd000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x72d3d73cd000
mmap(0x72d3d73ce000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x72d3d73ce000
mmap(0x72d3d73cf000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x72d3d73cf000
close(3)                                = 0
mprotect(0x72d3d73cf000, 4096, PROT_READ) = 0
mmap(NULL, 1052672, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x72d3d6eff000
write(1, "Allocation successful!\n", 23Allocation successful!
) = 23
write(1, "ptr1: 0x72d3d6eff040 (100 bytes)"..., 33ptr1: 0x72d3d6eff040 (100 bytes)
) = 33
write(1, "ptr2: 0x72d3d6eff0bc (200 bytes)"..., 33ptr2: 0x72d3d6eff0bc (200 bytes)
) = 33
write(1, "\n", 1
)                       = 1
write(1, "1 ptr0x72d3d6eff040\n", 201 ptr0x72d3d6eff040
)   = 20
write(1, "2 ptr0x72d3d6eff0d8\n", 202 ptr0x72d3d6eff0d8
)   = 20
write(1, "3 ptr0x72d3d6eff170\n", 203 ptr0x72d3d6eff170
)   = 20
write(1, "4 ptr0x72d3d6eff208\n", 204 ptr0x72d3d6eff208
)   = 20
write(1, "5 ptr0x72d3d6eff2a0\n", 205 ptr0x72d3d6eff2a0
)   = 20
write(1, "6 ptr0x72d3d6eff338\n", 206 ptr0x72d3d6eff338
)   = 20
write(1, "7 ptr0x72d3d6eff3d0\n", 207 ptr0x72d3d6eff3d0
)   = 20
write(1, "8 ptr0x72d3d6eff468\n", 208 ptr0x72d3d6eff468
)   = 20
write(1, "9 ptr0x72d3d6eff500\n", 209 ptr0x72d3d6eff500
)   = 20
write(1, "10 ptr0x72d3d6eff598\n", 2110 ptr0x72d3d6eff598
)  = 21
```

```
write(1, "11 ptr0x72d3d6eff630\n", 2111 ptr0x72d3d6eff630
)  = 21
write(1, "12 ptr0x72d3d6eff6c8\n", 2112 ptr0x72d3d6eff6c8
)  = 21
write(1, "13 ptr0x72d3d6eff760\n", 2113 ptr0x72d3d6eff760
)  = 21
write(1, "14 ptr0x72d3d6eff7f8\n", 2114 ptr0x72d3d6eff7f8
)  = 21
write(1, "15 ptr0x72d3d6eff890\n", 2115 ptr0x72d3d6eff890
)  = 21
write(1, "16 ptr0x72d3d6eff928\n", 2116 ptr0x72d3d6eff928
)  = 21
write(1, "17 ptr0x72d3d6eff9c0\n", 2117 ptr0x72d3d6eff9c0
)  = 21
write(1, "18 ptr0x72d3d6effa58\n", 2118 ptr0x72d3d6effa58
)  = 21
write(1, "19 ptr0x72d3d6effaf0\n", 2119 ptr0x72d3d6effaf0
)  = 21
write(1, "20 ptr0x72d3d6effb88\n", 2120 ptr0x72d3d6effb88
)  = 21
write(1, "21 ptr0x72d3d6effc20\n", 2121 ptr0x72d3d6effc20
)  = 21
write(1, "22 ptr0x72d3d6effcb8\n", 2122 ptr0x72d3d6effcb8
)  = 21
write(1, "23 ptr0x72d3d6effd50\n", 2123 ptr0x72d3d6effd50
)  = 21
write(1, "24 ptr0x72d3d6effde8\n", 2124 ptr0x72d3d6effde8
)  = 21
write(1, "25 ptr0x72d3d6effe80\n", 2125 ptr0x72d3d6effe80
)  = 21
write(1, "26 ptr0x72d3d6efff18\n", 2126 ptr0x72d3d6efff18
)  = 21
write(1, "27 ptr0x72d3d6efffb0\n", 2127 ptr0x72d3d6efffb0
)  = 21
write(1, "28 ptr0x72d3d6f00048\n", 2128 ptr0x72d3d6f00048
)  = 21
write(1, "29 ptr0x72d3d6f000e0\n", 2129 ptr0x72d3d6f000e0
)  = 21
write(1, "30 ptr0x72d3d6f00178\n", 2130 ptr0x72d3d6f00178
)  = 21
write(1, "31 ptr0x72d3d6f00210\n", 2131 ptr0x72d3d6f00210
)  = 21
write(1, "32 ptr0x72d3d6f002a8\n", 2132 ptr0x72d3d6f002a8
)  = 21
write(1, "33 ptr0x72d3d6f00340\n", 2133 ptr0x72d3d6f00340
)  = 21
write(1, "34 ptr0x72d3d6f003d8\n", 2134 ptr0x72d3d6f003d8
)  = 21
write(1, "35 ptr0x72d3d6f00470\n", 2135 ptr0x72d3d6f00470
```

```
)  = 21
write(1, "36 ptr0x72d3d6f00508\n", 2136 ptr0x72d3d6f00508
)  = 21
write(1, "37 ptr0x72d3d6f005a0\n", 2137 ptr0x72d3d6f005a0
)  = 21
write(1, "38 ptr0x72d3d6f00638\n", 2138 ptr0x72d3d6f00638
)  = 21
write(1, "39 ptr0x72d3d6f006d0\n", 2139 ptr0x72d3d6f006d0
)  = 21
write(1, "40 ptr0x72d3d6f00768\n", 2140 ptr0x72d3d6f00768
)  = 21
write(1, "41 ptr0x72d3d6f00800\n", 2141 ptr0x72d3d6f00800
)  = 21
write(1, "42 ptr0x72d3d6f00898\n", 2142 ptr0x72d3d6f00898
)  = 21
write(1, "43 ptr0x72d3d6f00930\n", 2143 ptr0x72d3d6f00930
)  = 21
write(1, "44 ptr0x72d3d6f009c8\n", 2144 ptr0x72d3d6f009c8
)  = 21
write(1, "45 ptr0x72d3d6f00a60\n", 2145 ptr0x72d3d6f00a60
)  = 21
write(1, "46 ptr0x72d3d6f00af8\n", 2146 ptr0x72d3d6f00af8
)  = 21
write(1, "47 ptr0x72d3d6f00b90\n", 2147 ptr0x72d3d6f00b90
)  = 21
write(1, "48 ptr0x72d3d6f00c28\n", 2148 ptr0x72d3d6f00c28
)  = 21
write(1, "49 ptr0x72d3d6f00cc0\n", 2149 ptr0x72d3d6f00cc0
)  = 21
write(1, "50 ptr0x72d3d6f00d58\n", 2150 ptr0x72d3d6f00d58
)  = 21
write(1, "51 ptr0x72d3d6f00df0\n", 2151 ptr0x72d3d6f00df0
)  = 21
write(1, "52 ptr0x72d3d6f00e88\n", 2152 ptr0x72d3d6f00e88
)  = 21
write(1, "53 ptr0x72d3d6f00f20\n", 2153 ptr0x72d3d6f00f20
)  = 21
write(1, "54 ptr0x72d3d6f00fb8\n", 2154 ptr0x72d3d6f00fb8
)  = 21
write(1, "55 ptr0x72d3d6f01050\n", 2155 ptr0x72d3d6f01050
)  = 21
write(1, "56 ptr0x72d3d6f010e8\n", 2156 ptr0x72d3d6f010e8
)  = 21
write(1, "57 ptr0x72d3d6f01180\n", 2157 ptr0x72d3d6f01180
)  = 21
write(1, "58 ptr0x72d3d6f01218\n", 2158 ptr0x72d3d6f01218
)  = 21
write(1, "59 ptr0x72d3d6f012b0\n", 2159 ptr0x72d3d6f012b0
)  = 21
```

```
write(1, "60 ptr0x72d3d6f01348\n", 2160 ptr0x72d3d6f01348
)  = 21
write(1, "61 ptr0x72d3d6f013e0\n", 2161 ptr0x72d3d6f013e0
)  = 21
write(1, "62 ptr0x72d3d6f01478\n", 2162 ptr0x72d3d6f01478
)  = 21
write(1, "63 ptr0x72d3d6f01510\n", 2163 ptr0x72d3d6f01510
)  = 21
write(1, "64 ptr0x72d3d6f015a8\n", 2164 ptr0x72d3d6f015a8
)  = 21
write(1, "65 ptr0x72d3d6f01640\n", 2165 ptr0x72d3d6f01640
)  = 21
write(1, "66 ptr0x72d3d6f016d8\n", 2166 ptr0x72d3d6f016d8
)  = 21
write(1, "67 ptr0x72d3d6f01770\n", 2167 ptr0x72d3d6f01770
)  = 21
write(1, "68 ptr0x72d3d6f01808\n", 2168 ptr0x72d3d6f01808
)  = 21
write(1, "69 ptr0x72d3d6f018a0\n", 2169 ptr0x72d3d6f018a0
)  = 21
write(1, "70 ptr0x72d3d6f01938\n", 2170 ptr0x72d3d6f01938
)  = 21
write(1, "71 ptr0x72d3d6f019d0\n", 2171 ptr0x72d3d6f019d0
)  = 21
write(1, "72 ptr0x72d3d6f01a68\n", 2172 ptr0x72d3d6f01a68
)  = 21
write(1, "73 ptr0x72d3d6f01b00\n", 2173 ptr0x72d3d6f01b00
)  = 21
write(1, "74 ptr0x72d3d6f01b98\n", 2174 ptr0x72d3d6f01b98
)  = 21
write(1, "75 ptr0x72d3d6f01c30\n", 2175 ptr0x72d3d6f01c30
)  = 21
write(1, "76 ptr0x72d3d6f01cc8\n", 2176 ptr0x72d3d6f01cc8
)  = 21
write(1, "77 ptr0x72d3d6f01d60\n", 2177 ptr0x72d3d6f01d60
)  = 21
write(1, "78 ptr0x72d3d6f01df8\n", 2178 ptr0x72d3d6f01df8
)  = 21
write(1, "79 ptr0x72d3d6f01e90\n", 2179 ptr0x72d3d6f01e90
)  = 21
write(1, "80 ptr0x72d3d6f01f28\n", 2180 ptr0x72d3d6f01f28
)  = 21
write(1, "81 ptr0x72d3d6f01fc0\n", 2181 ptr0x72d3d6f01fc0
)  = 21
write(1, "82 ptr0x72d3d6f02058\n", 2182 ptr0x72d3d6f02058
)  = 21
write(1, "83 ptr0x72d3d6f020f0\n", 2183 ptr0x72d3d6f020f0
)  = 21
write(1, "84 ptr0x72d3d6f02188\n", 2184 ptr0x72d3d6f02188
```

```
)  = 21
write(1, "85 ptr0x72d3d6f02220\n", 2185 ptr0x72d3d6f02220
)  = 21
write(1, "86 ptr0x72d3d6f022b8\n", 2186 ptr0x72d3d6f022b8
)  = 21
write(1, "87 ptr0x72d3d6f02350\n", 2187 ptr0x72d3d6f02350
)  = 21
write(1, "88 ptr0x72d3d6f023e8\n", 2188 ptr0x72d3d6f023e8
)  = 21
write(1, "89 ptr0x72d3d6f02480\n", 2189 ptr0x72d3d6f02480
)  = 21
write(1, "90 ptr0x72d3d6f02518\n", 2190 ptr0x72d3d6f02518
)  = 21
write(1, "91 ptr0x72d3d6f025b0\n", 2191 ptr0x72d3d6f025b0
)  = 21
write(1, "92 ptr0x72d3d6f02648\n", 2192 ptr0x72d3d6f02648
)  = 21
write(1, "93 ptr0x72d3d6f026e0\n", 2193 ptr0x72d3d6f026e0
)  = 21
write(1, "94 ptr0x72d3d6f02778\n", 2194 ptr0x72d3d6f02778
)  = 21
write(1, "95 ptr0x72d3d6f02810\n", 2195 ptr0x72d3d6f02810
)  = 21
write(1, "96 ptr0x72d3d6f028a8\n", 2196 ptr0x72d3d6f028a8
)  = 21
write(1, "97 ptr0x72d3d6f02940\n", 2197 ptr0x72d3d6f02940
)  = 21
write(1, "98 ptr0x72d3d6f029d8\n", 2198 ptr0x72d3d6f029d8
)  = 21
write(1, "99 ptr0x72d3d6f02a70\n", 2199 ptr0x72d3d6f02a70
)  = 21
write(1, "100 ptr0x72d3d6f02b08\n", 22100 ptr0x72d3d6f02b08
) = 22
write(1, "\n", 1
)                       = 1
write(1, "No memory leaks detected!\n", 26No memory leaks detected!
) = 26
write(1, "\n", 1
)                       = 1
brk(0x5ea6f551b000)                     = 0x5ea6f551b000
write(1, "Memory exhausted after allocatin"..., 47Memory exhausted after allocating
882944 bytes
) = 47
write(1, "Allocated 100 bytes at 0x72d3d6e"..., 38Allocated 100 bytes at 0x72d3d6eff040
) = 38
write(1, "Reallocated 100 bytes at 0x72d3d"..., 40Reallocated 100 bytes at
0x72d3d6eff040
) = 40
munmap(0x72d3d6eff000, 1052672)         = 0
```

```
munmap(0x72d3d73cc000, 16440)           = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Программа является эффективным инструментом для тестирования и проверки корректности работы аллокаторов памяти. Она позволяет выявить ошибки в реализации аллокаторов, такие как утечки памяти, некорректное выделение или освобождение памяти, а также проблемы с повторным использованием освобожденных блоков. Благодаря использованию динамической загрузки библиотек, программа может быть легко адаптирована для тестирования различных реализаций аллокаторов.