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-214Б-23

Студент: Миронов Д. А.

Преподаватель: Бахарев В.Д. (ФИИТ)

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 13.12.24

Москва, 2024

**Постановка задачи**

**Вариант 5.**

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). Аллокаторы – метод Мак-Кьюзика-Кэрелса и алгоритм двойников.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* void \*mmap(void addr, size\_t length, int prot, int flags, int fd, off\_t offset); – выполняет отображение файла или устройства на память.
* int munmap(void addr, size\_t length); – удаляет отображение файла или устройства на память.

Программа main в функции LoadLibrary загружает динамическую библиотеку по указанному пути используя dlopen. Если библиотеку загрузить не удалось, выводится сообщение об ошибке и указателям на функции присвоены указатели на функции по умолчанию, оборачивающие mmap и munmap. Если загрузить библиотеку удалось, то программа пытается найти в библиотеке соответствующие функции и присвоить их указателям из структуры Allocator. Если функции нет, то соответствующему указателю на функцию присвоен указатель на функцию оборачивающую mmap или munmap. Функция LoadLibrary вызывается с параметром argv[1], указывающем путь до библиотеки.

Библиотека MucKus реализует аллокатор на основании метода Мак-Кьюзика-Кэрелса. В этом аллокаторе мы будем хранить массив классов, каждый из которых будет представлять собой список свободных блоков определённого размера. Инициализация аллокатора (allocator\_create): аллокатор принимает на вход память (memory) и её размер (size). Аллокатор инициализирует массив размеров классов и массив самих классов. Запрос памяти (allocator\_alloc): аллокатор находит класс с минимальным размером, удовлетворяющим запросу и ищет свободный блок в массиве классов, возвращая его. Если такого блока нет, аллокатор выделяет блок из глобальной памяти. Освобождение памяти (allocator\_free): проверяет, принадлежит ли переданный указатель данному аллокатору, затем находит размер класса, к которому принадлежит данный блок и добавляет этот блок в нужный список массива классов. Очистка аллокатора (allocator\_destroy): очищает список свободных блоков.

Библиотека dsDvoiniki реализует аллокатор на основании метода двойников. В этом аллокаторе память выделяется блоками, размером 2^n. Инициализация аллокатора (allocator\_create): аллокатор принимает на вход память (memory) и её размер (size), выделяет часть памяти для структуры аллокатора (Allocator) и битовой карты, предварительно рассчитав её размер по количеству блоков памяти, которая является маркером занятых блоков. Битовая карта заполняется нулями. Запрос памяти (allocator\_alloc): запрашиваемый размер выравнивается до ближайшей степени двойки, если это необходимо. Аллокатор подсчитывает индекс блока в битовой карте, затем проверяет его на незанятость, а также проверяет на незанятость все смежные с ним блоки. Если все свободно, то блок помечается занятым и возвращается адрес на него. Освобождение памяти (allocator\_free): вычисляет сдвиг, затем считает индекс блока, перебирая все возможные размеры блоков, начиная с наименьшего. Проверяет занятость этого блока и освобождает его. Очистка аллокатора (allocator\_destroy): зануляет указатель на память и битовую катру, обнуляет размеры аллокатора и битовой карты.

**Код программы**

**MacKus.c**

#include <stdlib.h>

#include <string.h>

#include <stdint.h>

#include <stdio.h>

#define MAX\_CLASSES 20

**typedef** **struct** Block {

**struct** Block\* next;

size\_t size;

} Block;

**typedef** **struct** Allocator {

Block\* free\_list[MAX\_CLASSES];

size\_t class\_sizes[MAX\_CLASSES];

**void**\* memory\_start;

size\_t memory\_size;

} Allocator;

**static** size\_t find\_class(size\_t size, size\_t\* class\_sizes, size\_t num\_classes) {

**for** (size\_t i = 0; i < num\_classes; i++) {

**if** (size <= class\_sizes[i]) {

**return** i;

}

}

**return** num\_classes; // Error case

}

Allocator\* allocator\_create(**void**\* **const** memory, **const** size\_t size) {

Allocator\* allocator = (Allocator\*)memory;

**if** (size < **sizeof**(Allocator)) **return** NULL;

allocator->memory\_start = (**char**\*)memory + **sizeof**(Allocator);

allocator->memory\_size = size - **sizeof**(Allocator);

// Инициализация классов размеров

size\_t block\_size = 16;

**for** (size\_t i = 0; i < MAX\_CLASSES; i++) {

allocator->class\_sizes[i] = block\_size;

allocator->free\_list[i] = NULL;

block\_size \*= 2;

}

**return** allocator;

}

**void** allocator\_destroy(Allocator\* **const** allocator) {

// Очистка списков свободных блоков

**for** (size\_t i = 0; i < MAX\_CLASSES; i++) {

allocator->free\_list[i] = NULL;

}

}

**void**\* allocator\_alloc(Allocator\* **const** allocator, **const** size\_t size) {

size\_t class\_index = find\_class(size, allocator->class\_sizes, MAX\_CLASSES);

**if** (class\_index >= MAX\_CLASSES) {

printf("tut\n");

**return** NULL;

}

Block\* block = allocator->free\_list[class\_index];

**if** (block) {

block->size = size;

allocator->free\_list[class\_index] = block->next;

**return** (**void**\*)block;

}

// Если свободных блоков нет, выделяем из общей памяти

size\_t block\_size = allocator->class\_sizes[class\_index];

**if** (allocator->memory\_size < block\_size) **return** NULL;

**void**\* memory = allocator->memory\_start;

allocator->memory\_start = (**char**\*)allocator->memory\_start + block\_size;

allocator->memory\_size -= block\_size;

**return** memory;

}

**void** allocator\_free(Allocator\* **const** allocator, **void**\* **const** memory) {

**if** (!memory) **return**;

uintptr\_t addr = (uintptr\_t)memory - (uintptr\_t)allocator;

**if** (addr >= allocator->memory\_size) **return**;

Block\* block = (Block\*)memory;

size\_t class\_index = 0;

size\_t block\_size = 0;

**for** (; class\_index < MAX\_CLASSES; class\_index++) {

block\_size = allocator->class\_sizes[class\_index];

**if** (block->size <= block\_size) {

**break**;

}

}

**if** (class\_index >= MAX\_CLASSES) **return**;

//Block\* block = (Block\*)memory;

block->next = allocator->free\_list[class\_index];

allocator->free\_list[class\_index] = block;

}

**dsDvoiniki.c**

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/mman.h>

**typedef** **struct** Allocator {

**void** \*memory;

size\_t size;

uint8\_t \*bitmap;

size\_t bitmap\_size;

} Allocator;

#define MIN\_BLOCK\_SIZE 16

**void** printBits(uint8\_t num) {

**for** (**int** i = 7; i >= 0; i--) {

printf("%d", (num >> i) & 1);

}

printf("|");

}

size\_t next\_power\_of\_two(size\_t size) {

size\_t power = 1;

**while** (power < size) {

power <<= 1;

}

**return** power;

}

size\_t get\_block\_index(size\_t block\_size, size\_t total\_size, size\_t offset) {

**return** (offset / block\_size) + (total\_size / block\_size) - 1;

}

Allocator\* allocator\_create(**void** \***const** memory, **const** size\_t size) {

**if** (size < MIN\_BLOCK\_SIZE) {

**return** NULL;

}

Allocator \*allocator = (Allocator \*)memory;

//printf("start addr: %p\n", memory);

allocator->memory = (uint8\_t \*)memory + **sizeof**(Allocator);

allocator->size = size - **sizeof**(Allocator);

size\_t block\_size = MIN\_BLOCK\_SIZE;

size\_t total\_blocks = 0;

**while** (block\_size <= allocator->size) {

total\_blocks += allocator->size / block\_size;

block\_size \*= 2;

}

//printf("Total blocks %lu\n", total\_blocks);

allocator->bitmap\_size = (total\_blocks + 7) / 8; // Размер в байтах

//printf("Bitmap size: %lu\n", allocator->bitmap\_size);

allocator->bitmap = (uint8\_t \*)allocator->memory;

allocator->memory = (uint8\_t \*)allocator->memory + allocator->bitmap\_size;

allocator->size -= allocator->bitmap\_size;

//printf("valid addr: %p\n", allocator->memory);

//printf("allocator size: %lu\n", allocator->size);

memset(allocator->bitmap, 0, allocator->bitmap\_size);

**return** allocator;

}

**void** allocator\_destroy(Allocator \***const** allocator) {

allocator->memory = NULL;

allocator->bitmap\_size = 0;

allocator->bitmap = NULL;

allocator->size = 0;

}

**void**\* allocator\_alloc(Allocator \***const** allocator, **const** size\_t size) {

//printf("Bitmap: ");

//printBits(allocator->bitmap[0]);

//printBits(allocator->bitmap[1]);

//printf("\n");

**if** (size == 0 || size > allocator->size) {

**return** NULL;

}

size\_t block\_size = next\_power\_of\_two(size);

**if** (block\_size < MIN\_BLOCK\_SIZE) {

block\_size = MIN\_BLOCK\_SIZE;

}

//printf("----ALLOC START----\n");

size\_t offset = 0;

size\_t bitCnt = allocator->bitmap\_size \* 8;

**while** (block\_size <= allocator->size) {

//printf("Block size: %lu\n", block\_size);

size\_t index = get\_block\_index(block\_size, allocator->size, offset);

size\_t originIndex = index;

//printf("index: %lu\n", index);

**int** isFree = 1;

**if** ((allocator->bitmap[index / 8] & (1 << (7 - (index % 8)))) == 0) {

//Check smaller

size\_t checkCnt = 2;

**while** (index \* 2 + 2 < bitCnt) {

index = index \* 2 + 1;

**for** (**int** i = 0; i < checkCnt; i++) {

size\_t ind = index + i;

// printf("check index: %lu\n", ind);

// printf("byte: ");

// printBits(allocator->bitmap[ind / 8]);

// printf(" mask: ");

// printBits((1 << (7 - (ind % 8))));

// printf("\n");

// printf("Res: %d\n", (allocator->bitmap[ind / 8] & (1 << (7 - (ind % 8)))));

**if** ((allocator->bitmap[ind / 8] & (1 << (7 - (ind % 8)))) != 0) {

isFree = 0;

// printf("ahahahahaha\n");

**break**;

}

}

**if** (!isFree) {

**break**;

}

checkCnt \*= 2;

}

//Check bigger

**if** (isFree) {

index = originIndex;

**while** (index > 0) {

index = index / 2;

//printf("Check index: %lu\n", index);

**if** ((allocator->bitmap[index / 8] & (1 << (7 - (index % 8)))) != 0) {

isFree = 0;

**break**;

}

}

}

}

**if** (isFree) {

allocator->bitmap[originIndex / 8] |= (1 << (7 - (originIndex % 8)));

// printf("tut\n");

// printf("Memory: <%p> ", (uint8\_t \*)allocator->memory);

// printf("Offset: %lu. ", offset);

// printf("Addr: <%p>\n", (uint8\_t \*)allocator->memory + offset);

// printf("----ALLOC ENDED----\n\n");

**return** (**void**\*)((uint8\_t \*)allocator->memory + offset);

}

offset += block\_size;

}

**return** NULL; // Не удалось найти свободный блок

}

**void** allocator\_free(Allocator \***const** allocator, **void**\* **const** memory) {

//printf("\n##### FREE STARTED #####\n");

**if** (memory == NULL) {

printf("Memory null\n");

**return**;

}

uint8\_t\* mem = (uint8\_t\*)memory;

uint8\_t\* memStart = (uint8\_t\*)allocator->memory;

size\_t offset = mem - memStart;

//printf("offset: %lu\n", offset);

//size\_t offset = (uint8\_t \*)memory - (uint8\_t \*)allocator->memory;

size\_t block\_size = MIN\_BLOCK\_SIZE;

**while** (block\_size <= allocator->size) {

size\_t index = get\_block\_index(block\_size, allocator->size, offset);

//printf("Index: %lu\n", index);

**if** ((allocator->bitmap[index / 8] & (1 << (7 - (index % 8)))) != 0) {

allocator->bitmap[index / 8] &= ~(1 << (7 - (index % 8)));

**return**;

}

block\_size \*= 2;

}

}

**main.c**

#include <stddef.h>

#include <stdlib.h>

#include <stdint.h>

#include <string.h>

#include <stdio.h>

#include <unistd.h>

#include <dlfcn.h>

#include <sys/mman.h>

#include <time.h>

#define MEMORY\_SIZE 1024\*1024\*1024

#define BLOCK\_SIZE 128

#define TESTS\_COUNT 100

**typedef** **struct** Allocator {

size\_t size;

**void** \*memory;

} Allocator;

Allocator \*(\*allocator\_create)(**void** \*addr, size\_t size);

**void** \*(\*mem\_alloc)(Allocator \*allocator, size\_t size);

**void** (\*mem\_free)(Allocator \*allocator, **void** \*ptr);

**void** (\*allocator\_destroy)(Allocator \*allocator);

Allocator\* defaultCreate(**void** \***const** memory, **const** size\_t size) {

Allocator \*allocator = (Allocator\*) mmap(NULL, **sizeof**(Allocator), PROT\_READ | PROT\_WRITE, MAP\_PRIVATE | MAP\_ANON, -1, 0);

**if** (allocator != MAP\_FAILED) {

allocator->size = size;

allocator->memory = memory;

}

**return** allocator;

}

**void** defaultDestroy(Allocator \***const** allocator) {

munmap(allocator->memory, allocator->size);

munmap(allocator, **sizeof**(Allocator));

}

**void**\* defaultAlloc(Allocator\* allocator, **const** size\_t size) {

**return** mmap(NULL, size, PROT\_READ | PROT\_WRITE, MAP\_PRIVATE | MAP\_ANON, -1, 0);

}

**void** defaultFree(Allocator\* allocator, **void** \***const** memory) {

munmap(memory, **sizeof**(memory));

}

**void** LoadLibrary(**char** \*path) {

**if** (path == NULL || path[0] == '\0') {

**char** message[] = "Invalid path!\n";

write(STDERR\_FILENO, message, **sizeof**(message));

allocator\_create = defaultCreate;

mem\_alloc = defaultAlloc;

mem\_free = defaultFree;

allocator\_destroy = defaultDestroy;

**return**;

}

**void** \*library = dlopen(path, RTLD\_LOCAL | RTLD\_NOW);

**if** (!library) {

**char** message[] = "Failed to load library!\n";

write(STDERR\_FILENO, message, **sizeof**(message));

allocator\_create = defaultCreate;

mem\_alloc = defaultAlloc;

mem\_free = defaultFree;

allocator\_destroy = defaultDestroy;

**return**;

}

allocator\_create = dlsym(library, "allocator\_create");

mem\_alloc = dlsym(library, "allocator\_alloc");

mem\_free = dlsym(library, "allocator\_free");

allocator\_destroy = dlsym(library, "allocator\_destroy");

**if** (!allocator\_create || !mem\_alloc || !mem\_free || !allocator\_destroy) {

**const** **char** msg[] = "Failed to load allocator functions!\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

dlclose(library);

**return**;

}

}

**double** TestAllocator(Allocator \*allocator) {

**int** allocCnt = 100, equality = 0, type, offset = 0;

**void** \*memArray[100];

//printf("----STARTING TEST ALLOCATOR----\n");

clock\_t begin = clock();

**while** (allocCnt > 0) {

type = rand() % 2;

**if** (type == 0) {

**int** iter = rand() % 5;

**if** (iter == 0) {

iter = 1;

}

**for** (**int** i = 0; i < iter; i++) {

ssize\_t size = BLOCK\_SIZE;

**void** \*block = mem\_alloc(allocator, size);

**if** (block == NULL) {

printf("Failed to allocate %zd bytes!\n", size);

}

memArray[i + offset] = block;

}

equality += iter;

offset += iter;

allocCnt -= iter;

//printf("Allocated %d blocks\n", iter);

} **else** {

**if** (equality <= 0) {

**continue**;

} **else** {

**int** iter = rand() % 5;

**if** (iter == 0) {

iter = 1;

}

**if** (iter > equality) {

iter = equality;

}

**for** (**int** i = 0; i < iter; i++) {

mem\_free(allocator, memArray[offset - i - 1]);

}

equality -= iter;

offset -= iter;

//printf("Freed %d blocks\n", iter);

}

}

}

clock\_t end = clock();

//printf("----ENDING TEST ALLOCATOR----\n\n");

**double** elapsed = (**double**)(end - begin) / CLOCKS\_PER\_SEC;

printf("Elapsed time: %f\n", elapsed);

**return** elapsed;

}

**int** main(**int** argc, **char** \*argv[]) {

**char** \*path = argv[1];

LoadLibrary(path);

size\_t memory\_size = MEMORY\_SIZE;

**void** \*memory = mmap(NULL, memory\_size, PROT\_READ | PROT\_WRITE, MAP\_PRIVATE | MAP\_ANON, -1, 0);

**if** (memory == MAP\_FAILED) {

perror("mmap failed");

**return** 1;

}

Allocator \*allocator = allocator\_create(memory, memory\_size);

**if** (!allocator) {

fprintf(stderr, "Allocator creation failed\n");

**return** 1;

}

**double** elapsed = 0;

**for** (**int** i = 0; i < TESTS\_COUNT; ++i) {

elapsed += TestAllocator(allocator);

}

printf("\nAVG TIME: %f\n", elapsed / TESTS\_COUNT);

}

**Протокол работы программы**

**Тестирование:**

$ ./main ./dsDvoiniki.so

Elapsed time: 0.000013

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000011

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000016

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000011

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000011

Elapsed time: 0.000009

Elapsed time: 0.000011

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000011

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000006

Elapsed time: 0.000007

Elapsed time: 0.000007

Elapsed time: 0.000009

Elapsed time: 0.000011

Elapsed time: 0.000010

Elapsed time: 0.000011

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000011

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000007

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000007

Elapsed time: 0.000007

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000007

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000011

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000011

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000012

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000011

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000009

AVG TIME: 0.000009

$ ./main ./newMC.so

Elapsed time: 0.000009

Elapsed time: 0.000012

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000011

Elapsed time: 0.000010

Elapsed time: 0.000011

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000011

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000011

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000013

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000012

Elapsed time: 0.000008

Elapsed time: 0.000011

Elapsed time: 0.000011

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000007

Elapsed time: 0.000008

Elapsed time: 0.000013

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000011

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000013

Elapsed time: 0.000008

Elapsed time: 0.000007

Elapsed time: 0.000010

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000007

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000014

Elapsed time: 0.000009

Elapsed time: 0.000011

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000007

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000009

Elapsed time: 0.000010

Elapsed time: 0.000008

Elapsed time: 0.000008

Elapsed time: 0.000009

Elapsed time: 0.000008

Elapsed time: 0.000011

Elapsed time: 0.000008

Elapsed time: 0.000007

Elapsed time: 0.000008

Elapsed time: 0.000015

Elapsed time: 0.000009

Elapsed time: 0.000009

Elapsed time: 0.000008

AVG TIME: 0.000010

**Strace:**

$ strace ./main ./dsDvoiniki.so

execve("./main", ["./main", "./dsDvoiniki.so"], 0x7ffd13607b48 /\* 28 vars \*/) = 0

brk(NULL) = 0x55c51300e000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd4c3ee2000

access("/etc/ld.so.preload", R\_OK) = -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=20335, ...}) = 0

mmap(NULL, 20335, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fd4c3edd000

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\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) = 0x7fd4c3ccb000

mmap(0x7fd4c3cf3000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fd4c3cf3000

mmap(0x7fd4c3e7b000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7fd4c3e7b000

mmap(0x7fd4c3eca000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7fd4c3eca000

mmap(0x7fd4c3ed0000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fd4c3ed0000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd4c3cc8000

arch\_prctl(ARCH\_SET\_FS, 0x7fd4c3cc8740) = 0

set\_tid\_address(0x7fd4c3cc8a10) = 1464

set\_robust\_list(0x7fd4c3cc8a20, 24) = 0

rseq(0x7fd4c3cc9060, 0x20, 0, 0x53053053) = 0

mprotect(0x7fd4c3eca000, 16384, PROT\_READ) = 0

mprotect(0x55c50cb20000, 4096, PROT\_READ) = 0

mprotect(0x7fd4c3f1a000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7fd4c3edd000, 20335) = 0

getrandom("\x52\x31\x43\xe3\xc6\x70\xa2\x3f", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55c51300e000

brk(0x55c51302f000) = 0x55c51302f000

openat(AT\_FDCWD, "./dsDvoiniki.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|0777, st\_size=16136, ...}) = 0

getcwd("/mnt/c/Users/begemot/ClionProjects/OS-labs/Lab4", 128) = 48

mmap(NULL, 16504, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fd4c3edd000

mmap(0x7fd4c3ede000, 4096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000) = 0x7fd4c3ede000

mmap(0x7fd4c3edf000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7fd4c3edf000

mmap(0x7fd4c3ee0000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7fd4c3ee0000

close(3) = 0

mprotect(0x7fd4c3ee0000, 4096, PROT\_READ) = 0

mmap(NULL, 1073741824, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd483cc8000

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4267500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4321500}) = 0

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

write(1, "Elapsed time: 0.000054\n", 23Elapsed time: 0.000054

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4461900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4497500}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4578300}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4610500}) = 0

write(1, "Elapsed time: 0.000032\n", 23Elapsed time: 0.000032

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4712100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4750600}) = 0

write(1, "Elapsed time: 0.000038\n", 23Elapsed time: 0.000038

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4807400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4841600}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4907200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=4944200}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5009700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5045800}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5110700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5144800}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5210100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5244100}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5299800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5334300}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5400800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5434900}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5488000}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5523600}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5581200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5617000}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5682300}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5717500}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5778700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5803600}) = 0

write(1, "Elapsed time: 0.000025\n", 23Elapsed time: 0.000025

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5862100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5897000}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5961700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=5994600}) = 0

write(1, "Elapsed time: 0.000033\n", 23Elapsed time: 0.000033

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6060500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6093600}) = 0

write(1, "Elapsed time: 0.000033\n", 23Elapsed time: 0.000033

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6155800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6190000}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6246700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6284000}) = 0

write(1, "Elapsed time: 0.000038\n", 23Elapsed time: 0.000038

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6348600}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6383000}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6447800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6484700}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6539200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6574400}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6631800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6667800}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6733900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6768900}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6839800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6883800}) = 0

write(1, "Elapsed time: 0.000044\n", 23Elapsed time: 0.000044

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=6960800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7001300}) = 0

write(1, "Elapsed time: 0.000041\n", 23Elapsed time: 0.000041

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7066100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7102600}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7168500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7203700}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7259900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7294300}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7360200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7396500}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7452900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7490100}) = 0

write(1, "Elapsed time: 0.000038\n", 23Elapsed time: 0.000038

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7545800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7582400}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7652800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7687500}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7748100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7784200}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7843100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7878400}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7946100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=7989300}) = 0

write(1, "Elapsed time: 0.000043\n", 23Elapsed time: 0.000043

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8054300}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8089200}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8152200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8185100}) = 0

write(1, "Elapsed time: 0.000033\n", 23Elapsed time: 0.000033

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8247700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8279300}) = 0

write(1, "Elapsed time: 0.000032\n", 23Elapsed time: 0.000032

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8336200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8370300}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8438100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8471600}) = 0

write(1, "Elapsed time: 0.000033\n", 23Elapsed time: 0.000033

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8536900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8573200}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8642200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8679500}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8743000}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8778800}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8847100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8891300}) = 0

write(1, "Elapsed time: 0.000044\n", 23Elapsed time: 0.000044

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8952800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=8993200}) = 0

write(1, "Elapsed time: 0.000041\n", 23Elapsed time: 0.000041

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9053900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9093900}) = 0

write(1, "Elapsed time: 0.000040\n", 23Elapsed time: 0.000040

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9166400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9207400}) = 0

write(1, "Elapsed time: 0.000041\n", 23Elapsed time: 0.000041

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9271300}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9308700}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9374500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9409600}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9468200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9503800}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9560700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9597500}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9654800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9691700}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9748500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9781000}) = 0

write(1, "Elapsed time: 0.000033\n", 23Elapsed time: 0.000033

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9845700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9879800}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9942400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=9975400}) = 0

write(1, "Elapsed time: 0.000033\n", 23Elapsed time: 0.000033

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10042100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10075700}) = 0

write(1, "Elapsed time: 0.000033\n", 23Elapsed time: 0.000033

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10133500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10169700}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10234000}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10268400}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10334700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10372100}) = 0

write(1, "Elapsed time: 0.000038\n", 23Elapsed time: 0.000038

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10440100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10476400}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10542000}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10579100}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10644500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10677800}) = 0

write(1, "Elapsed time: 0.000033\n", 23Elapsed time: 0.000033

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10743100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10789900}) = 0

write(1, "Elapsed time: 0.000046\n", 23Elapsed time: 0.000046

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10856000}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10897600}) = 0

write(1, "Elapsed time: 0.000041\n", 23Elapsed time: 0.000041

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10953400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=10987400}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11043200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11078300}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11149100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11186700}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11264400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11307500}) = 0

write(1, "Elapsed time: 0.000043\n", 23Elapsed time: 0.000043

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11365600}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11402400}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11459400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11496300}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11549500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11583500}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11648800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11683100}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11740900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11778400}) = 0

write(1, "Elapsed time: 0.000038\n", 23Elapsed time: 0.000038

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11837600}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11871700}) = 0

write(1, "Elapsed time: 0.000034\n", 23Elapsed time: 0.000034

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11928600}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=11960800}) = 0

write(1, "Elapsed time: 0.000032\n", 23Elapsed time: 0.000032

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12031900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12067800}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12133800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12168700}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12236000}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12272100}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12344100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12380800}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12436900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12474100}) = 0

write(1, "Elapsed time: 0.000038\n", 23Elapsed time: 0.000038

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12538400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12574800}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12639900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12675400}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12741900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12777300}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12841000}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12877100}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12950700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=12987200}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13053600}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13089400}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13151100}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13187100}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13240900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13276400}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13353700}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13390000}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13455900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13491600}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13547500}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13585700}) = 0

write(1, "Elapsed time: 0.000038\n", 23Elapsed time: 0.000038

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13651600}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13687200}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13741400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13776300}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13831400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13868900}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13944400}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=13982100}) = 0

write(1, "Elapsed time: 0.000038\n", 23Elapsed time: 0.000038

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=14049800}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=14086700}) = 0

write(1, "Elapsed time: 0.000037\n", 23Elapsed time: 0.000037

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=14142200}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=14178800}) = 0

write(1, "Elapsed time: 0.000036\n", 23Elapsed time: 0.000036

) = 23

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=14234900}) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=14269000}) = 0

write(1, "Elapsed time: 0.000035\n", 23Elapsed time: 0.000035

) = 23

write(1, "\nAVG TIME: 0.000036\n", 20

AVG TIME: 0.000036

) = 20

exit\_group(0) = ?

+++ exited with 0 +++

|  |  |  |
| --- | --- | --- |
| **Объем памяти (байты)** | **MacKus аллокатор (мс)** | **dsDvoiniki аллокатор (мс)** |
| 1 Кб | 0.000004 | 0.000009 |
| 4 Кб | 0.000009 | 0.000009 |
| 16 Кб | 0.000014 | 0.000010 |
| 64 Кб | 0.000032 | 0.000015 |
| 256 Кб | 0.000102 | 0.000033 |
| 1 Мб | 0.000104 | 0.000107 |
| 4 Мб | 0.000143 | 0.000385 |

**Вывод**

Общее время выполнения: аллокатор, работающий по алгоритму двойников, значительно быстрее выделяет и освобождает память на достаточно большом промежутке значений, чем McKusick-Karels allocator. Изменением параметра MIN\_BLOCK\_SIZE время на больших объемах можно сократить на порядки. При настройке аллокатора двойников под конкретные объемы памяти, его использование будет выгоднее.

Потери памяти: внутренние потери у двух аллокаторов примерно одинаковые, так как оба округляют размер выделяемой памяти. Внешние потери у алгоритма Мак-Кьюзика-Кэрелса могут быть выше, так как при большом количестве мелких выделений, небольшие свободные блоки становятся недостаточными для последующих больших запросов, в то время как у алгоритма двойников, освобожденные блоки сливаются в блок более большого размера.