ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

Дисциплина:

«Операционные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

Выполнили:

Нгуен Хонг Хань N3249

_____

(подпись)

Проверил:

Савков Сергей Витальевич

_____

(подпись)

Санкт-Петербург

2022г.

Задание

Протестировать функцию malloc/free и построить график зависимости времени выделения от размера запрашиваемой памяти. Либо винда, либо линукс.

Сложный (или)

1. Сравнить с другими малоками
2. Тестировать на живом процессе

1. Теория

1.1. malloc()/free()

- Функция malloc() возвращает указатель void на выделенное пространство или значение NULL, если памяти недостаточно.
- Функция malloc выделяет блок памяти размером не менее size байтов. Размер блока может превышать size байтов из-за дополнительных затрат места на хранение информации о выравнивании и обслуживании.

1.2. new/delete

- New - пытается выделить и инициализировать объект или массив объектов указанного типа или заполнителя и возвращает подходящий типизированный ненулевой указатель на объект (или на исходный объект массива).

1.3. sbrk()

- Функция sbrk() используется для изменения объема памяти, выделенной для вызывающего процесса. Изменение вносится путем сброса значения прерывания процесса и выделения соответствующего объема памяти. Объем выделенного пространства увеличивается по мере увеличения значения разрыва.

1.4. Сравнения

Feature	new / delete	malloc / free	sbrk
Memory allocated from	'Free Store'	'Heap'	‘Heap’
Returns	Fully typed pointer	void*	The prior break value
On failure	Throws (never returns NULL)	Returns NULL	(void *)−1
Required size	Calculated by compiler	Must be specified in bytes	Are rounded up for alignment with eight-byte boundaries.
Reallocating	Not handled intuitively	Simple (no copy constructor)	Yes
Call of reverse	Implementation defined	No	
Low memory cases	Can add a new memory allocator	Not handled by user code	Return error if The data segment size limit was exceeded.
Overridable	Yes	No	Yes
Use of constructor / destructor	Yes	No	Yes

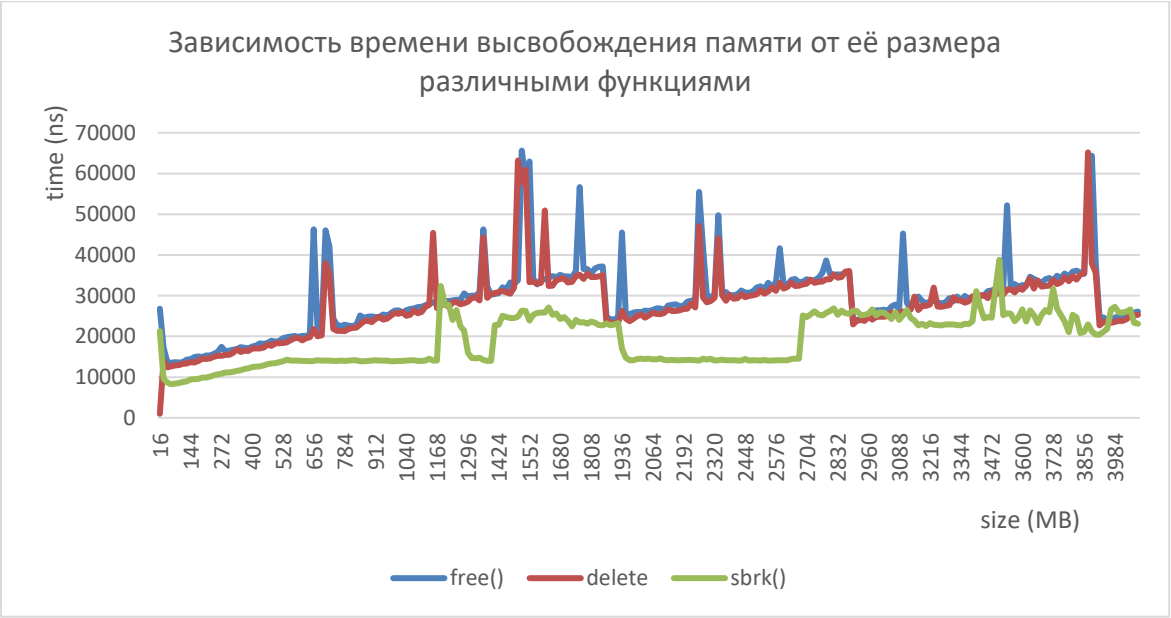
2. Код программы

```
#include <chrono>
#include <fstream>
#include <stdlib.h>
#include <linux/kernel.h>
#include <unistd.h>

void* callAlocator(int num, size_t size){
    if (num == 1){
        return malloc (sizeof(int) * size);
    }
```

```
    if (num == 2){
        return new int[size];
    }

    if (num == 3){
        return sbrk(sizeof(int) * size);
    }
    return NULL;
}

void callFree(int num, void *ptr){
    if (num == 1){
        free(ptr);
        return;
    }

    if (num == 2){
        delete ptr;
        return;
    }

}

int main () {
    std::ofstream resfile;
    resfile.open ("result1.txt");

    std::chrono::steady_clock::time_point begin, end;


    for (size_t i = 0; i < 1073741824; i = i + 1024*4096){
        resfile << i*4 << " ";
        for (int type = 1; type <= 3; type ++){

            begin = std::chrono::steady_clock::now();
            void* ptr = callAlocator(type, i);
            end = std::chrono::steady_clock::now();
            resfile << std::chrono::duration_cast<std::chrono::nanoseconds> (end -
begin).count() << " ";

            begin = std::chrono::steady_clock::now();
            callFree(type, ptr);
            // ptr = sbrk((-1)*sizeof(int) * i);
            end = std::chrono::steady_clock::now();
            resfile << std::chrono::duration_cast<std::chrono::nanoseconds> (end -
begin).count() << " ";
        }
        resfile << std::endl;
    }

    resfile.close();
    return 0;
}
```

3. **Протестировать функцию malloc/free и построить график зависимости времени выделения от размера запрашиваемой памяти на линусе.**

4. **Сравнить с другими малоками (new/delete, sbrk)**



5. Вывод

В ходе лабораторной работы была разработана программа на C++ для тестирования различных функций выделения памяти, а также была проанализирована их работа, а именно скорость относительно размера блока выделяемой памяти.