Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

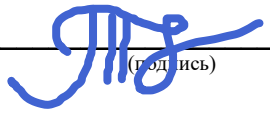
«Операционные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

«Аллокаторы памяти»

Выполнил:

Рядовой Т.С., студент группы N3252



(подпись)

Проверил:

Чебунин Константин Олегович

(отметка о выполнении)

(подпись)

Санкт-Петербург

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – протестировать функции malloc/free.

В обычном варианте:

- Тест;
- Построить график зависимости времени выделения от размера запрашиваемой памяти.

В усложненном варианте:

- Аналогичная работа с другими аллокаторами.

1 ОБЫЧНЫЙ ВАРИАНТ ЛАБОРАТОРНОЙ

1.1 Задание

Тестирование malloc/free на различных объемах памяти.

1.2 Код

Листинг 1 – Код malloc/free на C

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

int main(int argc, char *argv[]) {
    char *memory = NULL;

    for (int i = 0; i <= 2000   ; i += 128){
        clock_t start = clock();
        for (int j = 0; j <= 1000; ++j){
            memory = (char*)malloc(1024*1024*i);
            free(memory);
        }
        clock_t end = clock();
        printf("%d mb in %f seconds\n", i, ((double)(end - start) /
CLOCKS_PER_SEC));
    }
    return 0;
}
```

Листинг 2 – Запуск и вывод

```
gcc malloc.c
./a.out

0 mb in 0.000040 seconds
128 mb in 0.000577 seconds
256 mb in 0.000751 seconds
384 mb in 0.001141 seconds
512 mb in 0.001659 seconds
640 mb in 0.002340 seconds
768 mb in 0.003087 seconds
896 mb in 0.003758 seconds
1024 mb in 0.003958 seconds
1152 mb in 0.004307 seconds
1280 mb in 0.004887 seconds
1408 mb in 0.005665 seconds
1536 mb in 0.004634 seconds
1664 mb in 0.003923 seconds
1792 mb in 0.003922 seconds
1920 mb in 0.003916 seconds
```

1.3 График

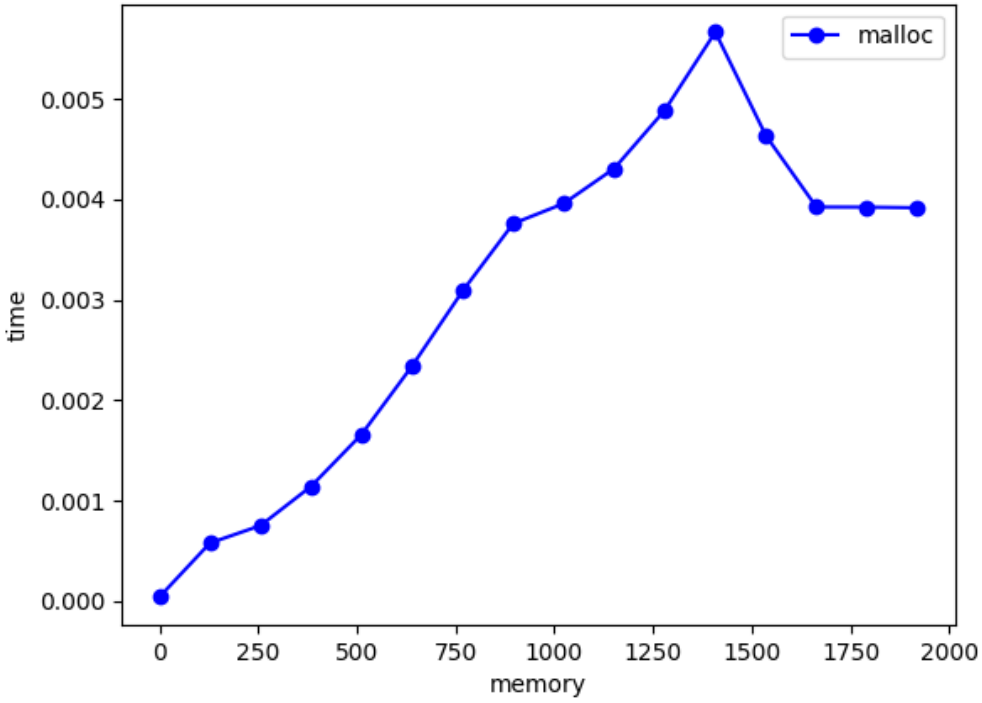


Рисунок 1 – График для функции malloc/free

2 УСИЛЕННЫЙ ВАРИАНТ ЛАБОРАТОРНОЙ

2.1 Задание

Проведем аналогичные тесты, используя функции calloc и tcmalloc.

2.2 Код

Листинг 3 – Код для calloc на C

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

int main(int argc, char *argv[]) {
    char *memory = NULL;

    for (int i = 0; i <= 2000   ; i += 128){
        clock_t start = clock();
        for (int j = 0; j <= 1000; ++j){
            memory = calloc(1024*1024*i, sizeof(char));
            free(memory);
        }
        clock_t end = clock();
        printf("%d mb in %f seconds\n", i, ((double)(end - start) /
CLOCKS_PER_SEC));
    }
    return 0;
}
```

Листинг 4 – Код для tcmalloc на C

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <gperftools/tcmalloc.h>

int main(int argc, char *argv[]) {
    char *memory = NULL;

    for (int i = 0; i <= 2000   ; i += 128){
        clock_t start = clock();
        for (int j = 0; j <= 1000; ++j){
            memory = (char*)tc_malloc(1024*1024*i);
            free(memory);
        }
        clock_t end = clock();
        printf("%d mb in %f seconds\n", i, ((double)(end - start) /
CLOCKS_PER_SEC));
    }
    return 0;
}
```

2.3 График

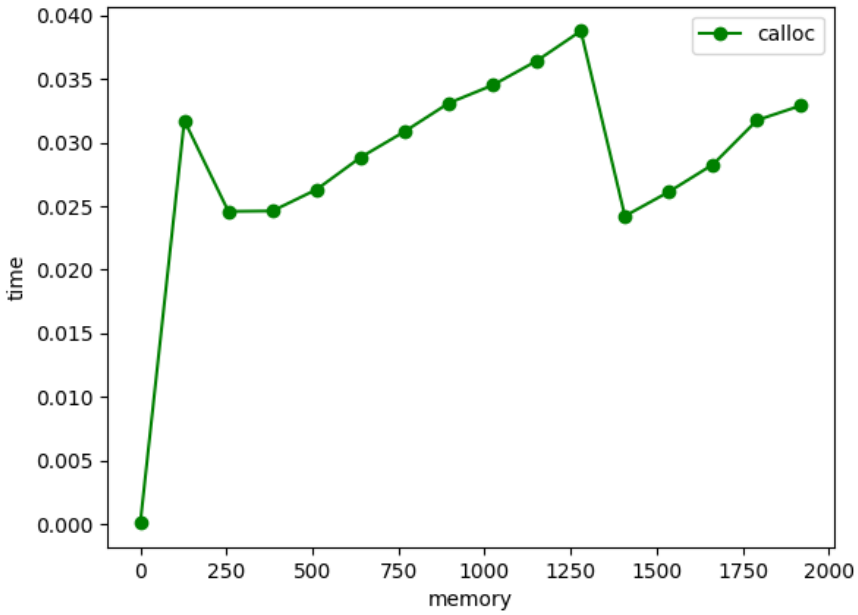


Рисунок 2 – График для функции calloc

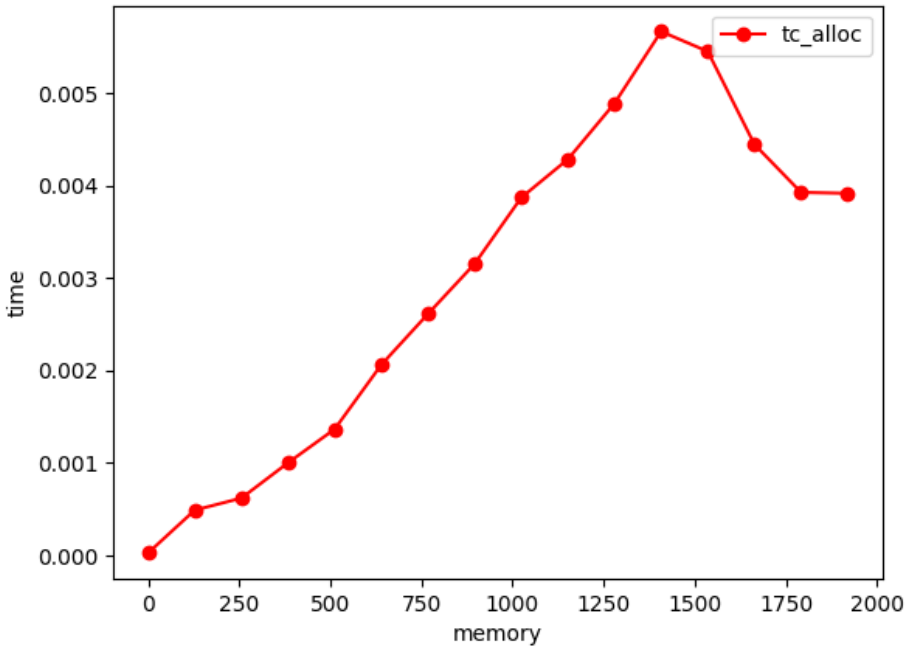


Рисунок 3 – График для функции tcmalloc

2.4 Сравнение

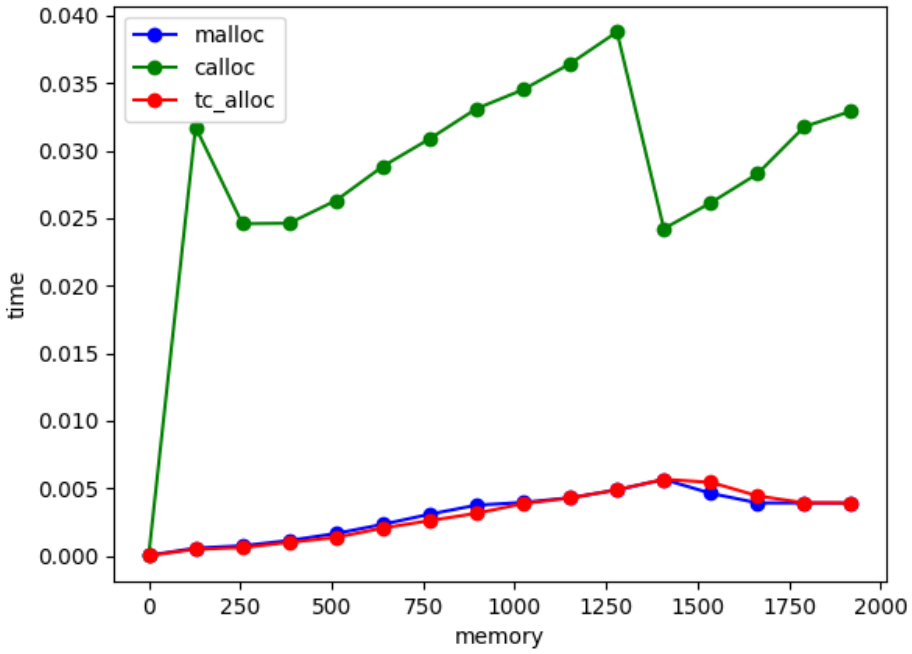


Рисунок 4 – Сравнение трех функций

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы мне удалось достигнуть поставленных целей:

Обычного варианта:

- Протестировать malloc/free на разном размере памяти;
- Построить график зависимости времени выделения памяти от ее размера.

Усложненного варианта:

- Протестировать 2 дополнительные функции;
- Сравнить их между собой и с malloc/free.

Из усложненного варианта видно, что malloc и tcmalloc различаются слабо. Функция calloc тратит больше времени на одинаковые блоки памяти.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://github.com/google/tcmalloc/blob/master/docs/overview.md
2. https://github.com/google/tcmalloc