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

«Тестирование функции malloc/free»

Выполнили:

Бардышев Артём Антонович,

студент группы N3246

_____

(подпись)

Суханкулиев Мухаммет,

студент группы N3246

_____

(подпись)

Проверил:

Савков Сергей Витальевич,

инженер

_____

(отметка о выполнении)

_____

(подпись)

Санкт-Петербург

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – протестировать функцию malloc/free и построить график зависимости времени выделения от размера запрашиваемой памяти.

Сложный вариант (или):

- Сравнить с другими маллоками;
- Тестировать на живом процессе.

1 ТЕСТИРОВАНИЕ ФУНКЦИЙ MALLOC/FREE

1.1 Описание программы

Для выполнения тестирования на Windows напишем программу на C.

- Выделение и освобождение памяти в программе выглядит как:

```
…void *ptr = malloc(size);…
free(ptr);…
```

- Для точных измерений времени выделения и освобождения памяти использован высокоточный таймер Windows — QueryPerformanceCounter. Он позволяет измерять время с точностью до наносекунд, что важно для таких операций, как malloc и free:

```
…#include <windows.h>…
QueryPerformanceFrequency(&frequency);…
      QueryPerformanceCounter(&start);…
      QueryPerformanceCounter(&end);…
      double malloc_time = (double)(end.QuadPart - start.QuadPart) * 1e9 /
      frequency.QuadPart;
      total_malloc_time += malloc_time;…
```

- Для повышения точности измерений и уменьшения влияния случайных факторов, операции усредняется по нескольким итерациям (`ITERATIONS = 5000`).

- Для тестирования различных размеров памяти используется логарифмическая последовательность (размеры увеличиваются в два раза от 1 байта до 1 ГБ):

```
for (size_t size = 1; size <= 1024 * 1024 * 1024; size *= 2) {…
```

- Операции выделения памяти с помощью malloc и освобождения с помощью free тестируются по отдельности, что позволяет точно измерить их индивидуальное время.

- В коде предусмотрена проверка на успешность выделения памяти с помощью malloc:

```
…if (!ptr) {
      perror("malloc failed");
      return 1;…
```

1.2 Анализ результатов

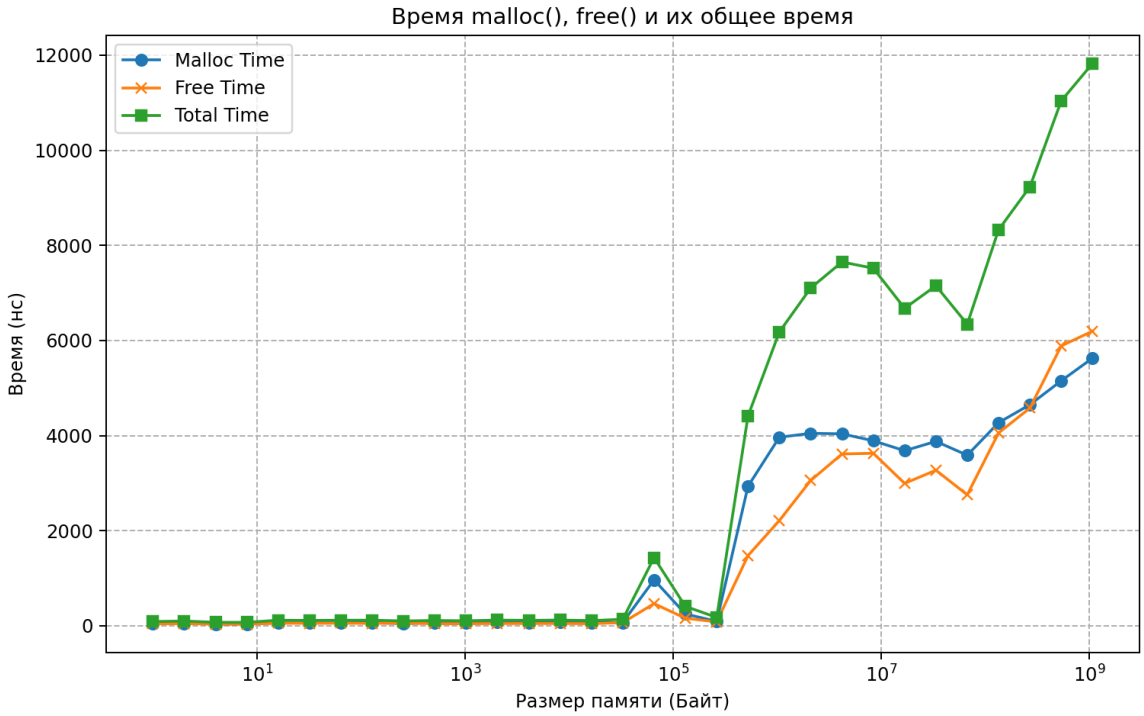

Рисунок 1 – Результаты тестирования malloc/free

По графикам мы наблюдаем, что время для malloc/free с увеличением размера памяти имеет экспоненциальный характер. Так же происходит резкий скачок во времени после ≈ 64МБ, в то время как на малых размерах памяти время почти не изменяется. Это связано с быстрым и эффективным управлением мелкими блоками памяти а после определённого порога система переходит на более сложные алгоритмы управления памятью, такие как виртуальная память, страничный обмен и дефрагментация.

2 СРАВНЕНИЯ С ДРУГИМИ МАЛЛОКАМИ

2.1 Сравнения с calloc и realloc

calloc (clear allocation): выделяет память для массива, инициализируя все байты нулями. Используется, когда требуется заранее обнуленная память.

realloc (reallocate): изменяет размер ранее выделенного блока памяти. Используется для расширения или сокращения динамической памяти без потери данных.

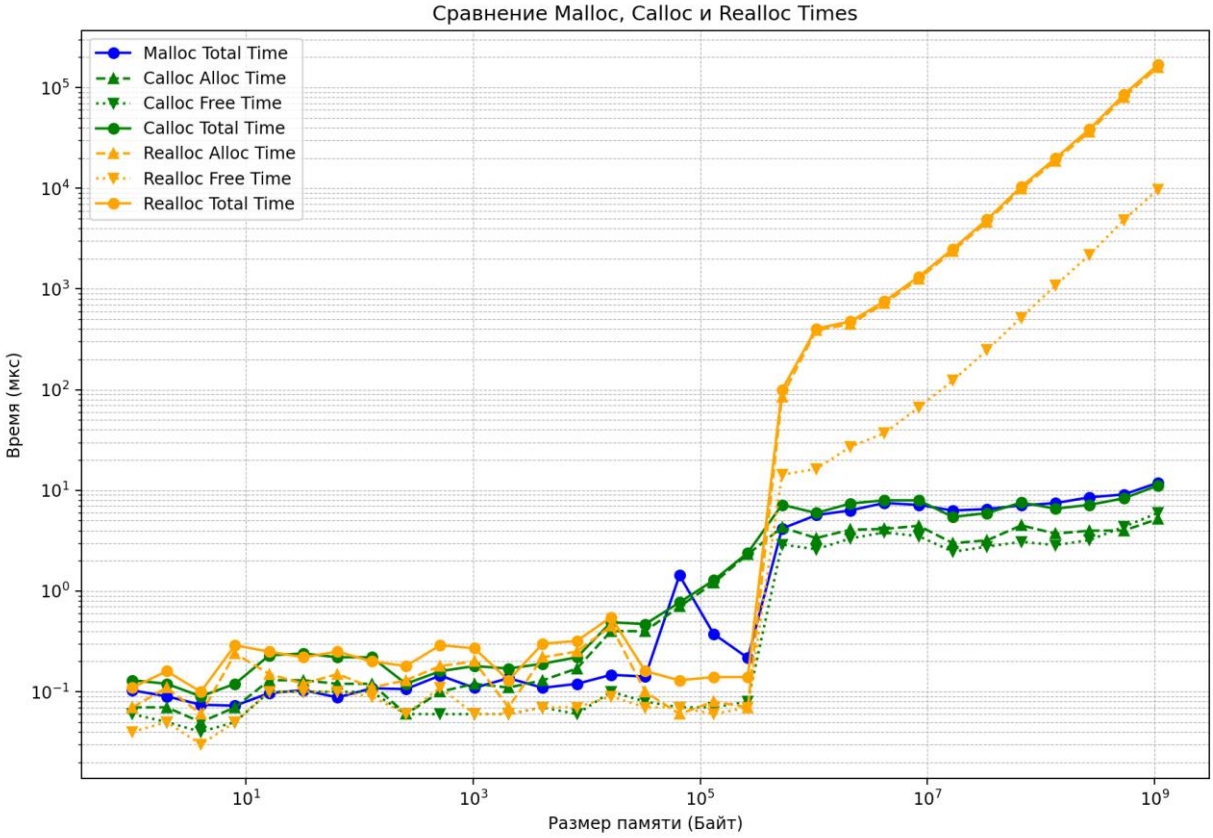

Рисунок 2 – Сравнение malloc, calloc и realloc

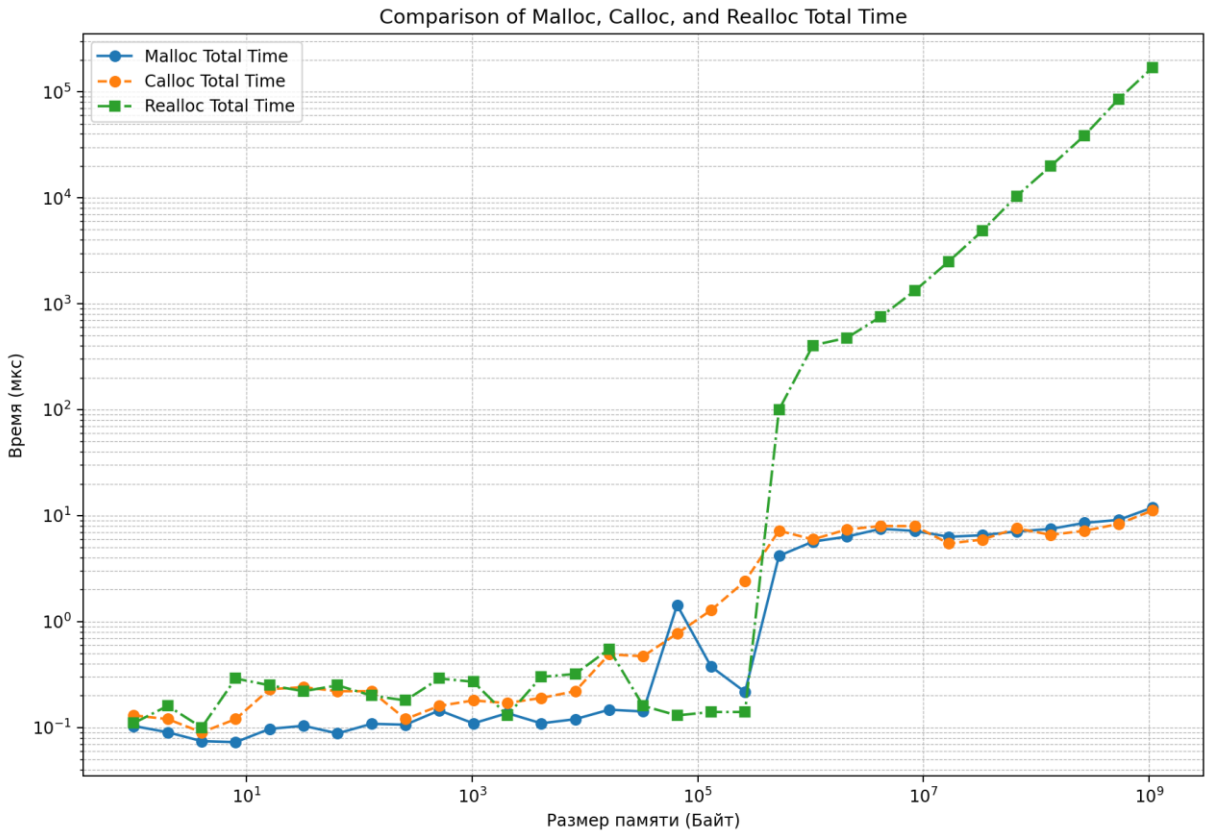

Рисунок 3 – Сравнение только total у malloc, calloc и realloc

Общий вывод:

malloc эффективнее и быстрее для больших блоков памяти, особенно если не требуется инициализация.

calloc подходит для случаев, когда важна инициализация памяти, но становится менее выгодным при работе с большими объемами.

realloc имеет существенные накладные расходы на больших объемах, и его применение следует тщательно планировать.

Если требуется максимальная производительность, особенно на больших объемах, стоит избегать частых перераспределений памяти (realloc) и использовать стратегии, которые минимизируют накладные расходы на инициализацию или копирование.

2.2 Сравнение malloc и VirtualAlloc

В отличие от стандартных функций malloc и free, VirtualAlloc предоставляет более низкоуровневое управление памятью и позволяет выделять большие блоки памяти с конкретным выравниванием или с особыми атрибутами.

Здесь мы выполняем те же операции, что и при тестировании malloc, только с VirtualAlloc:

```
…void *ptr = VirtualAlloc(NULL, size, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);…
VirtualFree(ptr, 0, MEM_RELEASE)…
```

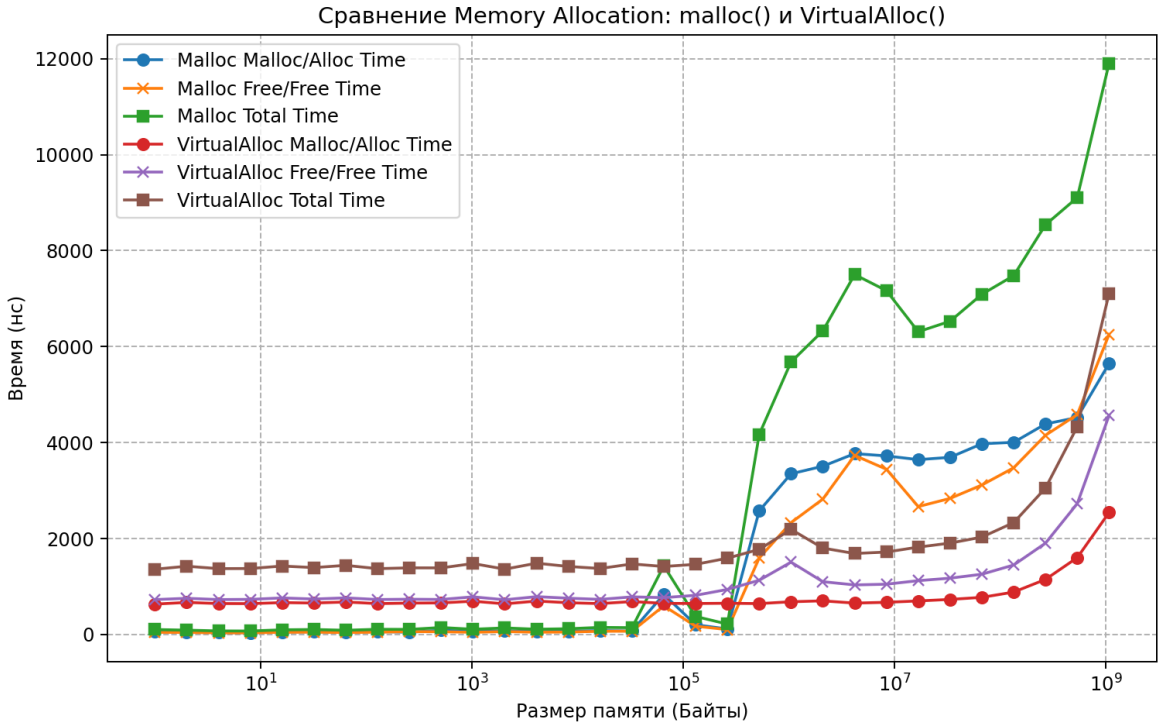

Рисунок 4 – Сравнение malloc и VirtualAlloc

На больших объемах памяти VirtualAlloc показывает большую стабильность в производительности по сравнению с malloc. Это связано с тем, что VirtualAlloc эффективно управляет большими объемами памяти, в то время как malloc начинает испытывать трудности с увеличением размера.

3 ТЕСТИРОВАНИЕ VIRTUALALLOC НА «ЖИВОМ» ПРОЦЕССЕ

3.1 Выполнение

В связи с замедлением тестирования (поскольку новый процесс будет работать в своем собственном виртуальном адресном пространстве, операционная система должна будет управлять этим пространством, что может замедлить доступ к памяти), установим `ITERATIONS 10`.

Создадим process.c, который запустит virtualalloc.exe, через новый, пустой процесс:

```
…CreateProcess(NULL, virtualalloc.exe, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi)…
```

По сути, операционная система создает новый процесс, в котором будет выполняться этот файл. В новом процессе запускается virtualalloc.exe, который начинает выделять и освобождать память с помощью VirtualAlloc и VirtualFree. В это время новый процесс работает в своем собственном виртуальном адресном пространстве и не мешает работе основного приложения.

То есть, создавая новый процесс, мы изолируем выполнение программы virtualalloc.exe от основного процесса.

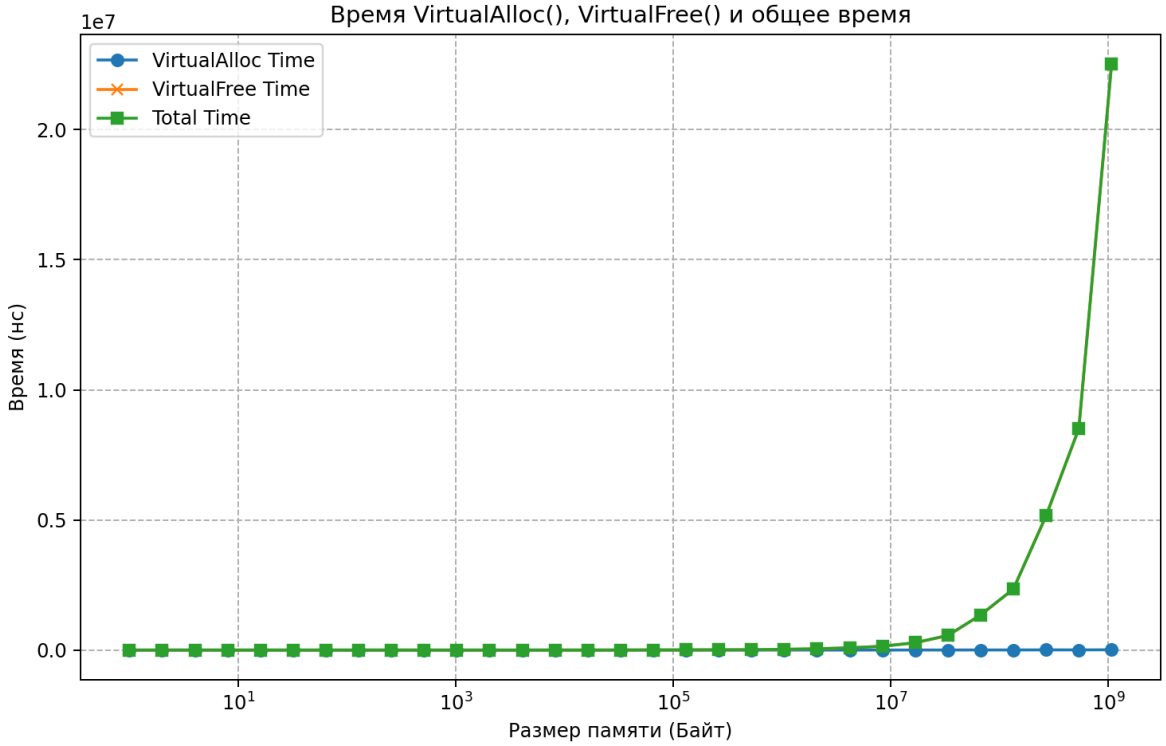

Рисунок 5 – Результаты тестирования VirtualAlloc на «живом» процессе

Стратегии управления памятью в Windows:

Выделение памяти (VirtualAlloc):

Память может быть резервирована или коммитирована без фактического выделения физической памяти. Это ускоряет процесс, так как физическая память используется только при активном обращении к ней (lazy allocation).

Освобождение памяти (VirtualFree):

Освобождение памяти требует обновления таблиц страниц и других структур данных. Это включает очистку и перераспределение ресурсов, что может занять больше времени, особенно для больших блоков памяти.

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы было проведено тестирование функций выделения и освобождения памяти — malloc/free и VirtualAlloc/VirtualFree. Результаты показали, что с увеличением размера выделяемой памяти время выполнения операций экспоненциально возрастает. В частности, для malloc/free на малых объемах памяти время операций практически не изменяется, но после достижения определенного порога происходит резкий скачок во времени. Это связано с переходом системы на более сложные методы управления памятью.

Использование calloc показало, что время выделения памяти с инициализацией (нулевыми значениями) обычно выше, чем у malloc, из-за дополнительной операции инициализации. Realloc продемонстрировала свои преимущества при изменении размера уже выделенного блока памяти, но в случае перемещения блока также может увеличиваться время выполнения.

Сравнение malloc и VirtualAlloc показало, что VirtualAlloc, в отличие от malloc, более эффективно управляет большими объемами памяти.

Тестирование VirtualAlloc в изолированном процессе подтвердило, что использование новых процессов позволяет минимизировать влияние на основное приложение и получать более точные результаты.

В целом, работа продемонстрировала важность выбора подходящих инструментов для работы с памятью в зависимости от задач, объёма данных и требований к производительности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Как устроено распределение памяти / Хабр
2. Функция VirtualAlloc (memoryapi.h) - Win32 apps | Microsoft Learn
3. Hack the Virtual Memory: malloc, the heap & the program break - Blog Holberton School
4. TCMalloc : Thread-Caching Malloc