Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №1

з дисципліни «Операційні системи»

Виконав:

студент гр. ІС-71

Андрощук Юрій

Перевірив:

Дифучин А. Ю.

Київ – 2020

**1.Опис алгоритму**

Був створений кастомний алокатор в якому реалізовані функціїї mem\_alloc, mem\_dump, mem\_realloc, mem\_free .

Представлення блока пам'яті у коді:

struct mem\_block

{

// memory location pointer

unsigned char\* ptr;

//size of the block which we will allocate

unsigned int size;

// pointer to the next memoru block

mem\_block\* next;

//pointer to the previous memory block

mem\_block\* prev;

};

Виділення пам’яті (mem\_alloc):

1)Вирівнюємо по довжині слова – 4 байти.

2)Намагаємось знайти вільний блок пам’яті з необхідним розміром

Знайшли =>повертаємо покажчик на блок якщо співпали по величині, якщо більша то обрізаємо і повертаємо обрізаний блок.

Не знайшли => створюємо новий – виділяємо пам'ять, встановлюємо покажчик на блок, блок стає хвостом області, повертаємо покажчик на блок

Виведення даних про пам’ять (mem\_dump):

Шукаємо 1 блок

Якщо first\_block == nullptr пам’ять пуста

Проходимось по блокам пам’яті і перевіряємо чи вони вільні.

Зміна існуючої пам’яті (mem\_realloc):

1.mem\_free

2.mem\_alloc

Дивимось на розмір блока – якщо більший(створюємо новий блок в кінці області),якщо ні(знаходимо блок що може підходити з початку області пам’яті ,або блок залишається на місці старого блока зі зміненим розміром беремо різницю і створюємо вільний блок що дорівнює різниці)

Звільнення пам’яті (mem\_free):

Блок видаляється зі списку зайнятих і потрапляє у список вільних блоків. У разі необхідності об’єднуємо блоки, що допомагає боротися з фрагментацією.

**2.Оцінка пошуку блоків**

Блоки перевіряються по списку. O(Pвільн), де Pвільн – кількість елементів у списку вільних блоків. Для зайнятих - O(Pзайн), де Pзайн – кількість елементів у списку зайнятих блоків.

**3.Оцінка службової пам’яті**

У даному випадку блок представляється структурою, тому зберігаємо її. Зберігати списки покажчиків на блоки, щоб відрізняти вільні від зайнятих.

**4.Недоліки та переваги**

* 1. Використання великих об’мів службової пам’яті
  2. При зміні розміру блоку, новий розмір буде більший за попередній + вільний простір справа, то в пам’яті виділиться новий блок і дані старого туди не перенесуться
  3. Існування фрагментації, доки ми вирівнюємо блоки до розмірів, кратних 4 байтам
  4. Вирівнюються адресів пам’яті на границю в 4 байти, це призводить до більшої універсальності при зміні уже створених блоків
  5. Швидкий пошук на наявність вільних блоків
  6. При звільненні блоку, він об’єднується з вільними блокам справа і зліва (якщо такі наявні). Це допомагає зменшити зовнішню фрагментацію пам’яті
  7. Метод mem\_dump дозволяє проглянути стан контрольованої пам’яті

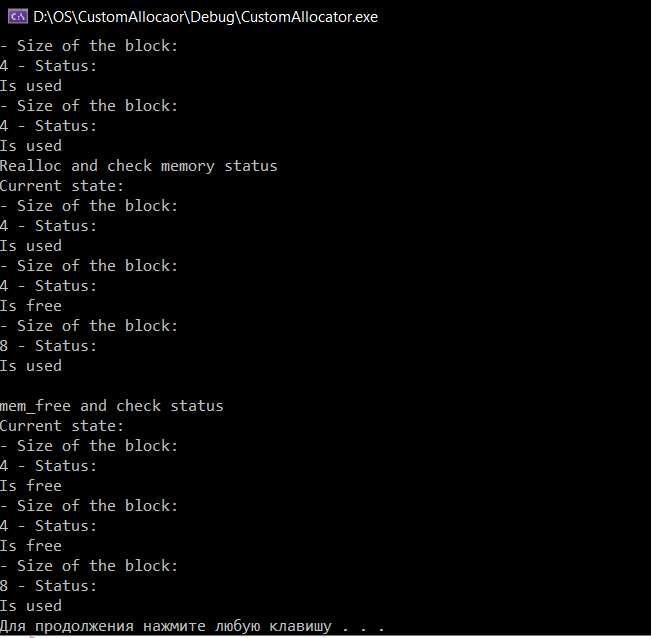
**5.Зменшення фрагментації пам’яті**

Об’єднання блоків під час звільнення пам’яті є способом боротьби з фрагментацією.

**6.Лістинг програми**

<https://github.com/Yura-Androshchuk/LabsOS>

**7.Приклад роботи**

.