МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 "Компьютерные науки и прикладная математика"

Кафедра 806 "Вычислительная математика и программирование"

Курсовой проект

По курсу «Операционные системы»

Студент: Лернер Ф. Л.

Группа: М8О-208Б-23

Преподаватель: Живалев Е. А.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Тема:** Аллокаторы памяти

**Цель работы:** Целью курсового проекта являлось приобретение практических навыков в:

* Целью данного проекта является исследование и сравнение двух различных аллокаторов памяти на языке C++ по ключевым характеристикам, таким как фактор использования, скорость выделения и освобождения блоков, а также простота использования. Проект направлен на понимание работы различных стратегий управления памятью и их влияние на производительность программ.

**Вариант:** 16. Необходимо сравнить два алгоритма аллокации: алгоритм Мак-Кьюзи-Кэрелса и блоки по 2 в степени n.

**Подробное описание каждого из исследуемых алгоритмов:**

1. McKusick Allocator

Основан на сегментации памяти на классы блоков фиксированного размера (16, 32, 64 байт и т.д.). Подходит для сценариев с небольшими запросами памяти.

1. Pow2 Allocator.

Делит память на блоки, размеры которых кратны степеням двойки. Подходит для сценариев с большими блоками, но может приводить к фрагментации.

**Задачи проекта:**

1. Разработка двух аллокаторов памяти: Реализовать два различных алгоритма аллокации памяти, например, "первый подходящий" (First-Fit) и "наилучший подходящий" (Best-Fit).

2. Создание интерфейса: Обеспечить наличие функций, аналогичных стандартным функциям malloc, free и (опционально) realloc.

3. Инициализация аллокаторов: Реализовать механизм инициализации аллокаторов с использованием стандартных средств выделения памяти в ядре.

4. Разработка стратегии тестирования: Определить и реализовать стратегию тестирования для оценки производительности аллокаторов по заданным характеристикам.

**Описание решения:**

решение разработано на языке C++ с использованием библиотеки pthread для работы с потоками. Основные компоненты системы:

Пример реализации:

#include "allocator\_mckusick.hpp"

#include <cstring>

#include <sys/mman.h>

#include <cassert>

#include <cmath>

AllocatorMcKusick::AllocatorMcKusick(size\_t total\_size) {

page\_size = 4096; // Размер страницы

memory\_pool = mmap(nullptr, total\_size, PROT\_READ | PROT\_WRITE, MAP\_PRIVATE | MAP\_ANONYMOUS, -1, 0);

assert(memory\_pool != MAP\_FAILED && "Failed to allocate memory pool");

block\_classes = 10; // 10 классов блоков (например, 16, 32, 64 и т.д.)

free\_lists.resize(block\_classes, nullptr);

}

AllocatorMcKusick::~AllocatorMcKusick() {

munmap(memory\_pool, page\_size \* block\_classes);

}

void\* AllocatorMcKusick::allocate(size\_t size) {

size\_t class\_idx = std::ceil(std::log2(size));

assert(class\_idx < block\_classes && "Requested size exceeds maximum class");

if (!free\_lists[class\_idx]) {

free\_lists[class\_idx] = static\_cast<char\*>(memory\_pool) + class\_idx \* page\_size;

}

void\* block = free\_lists[class\_idx];

free\_lists[class\_idx] = static\_cast<void\*>(static\_cast<char\*>(block) + (1 << class\_idx));

return block;

}

void AllocatorMcKusick::deallocate(void\* ptr) {

// Простая реализация: добавление блока обратно в список

memset(ptr, 0, sizeof(ptr));

**Данные из терминала:**

$ ./benchmark\_allocators

Running simplified tests...

=== test\_usage\_factor\_simple ===

McKusick usage factor: 0.00122

Pow2 usage factor: 0.00122

=== test\_alloc\_dealloc\_simple ===

McKusick: allocated & deallocated 10 blocks

Pow2: allocated & deallocated 10 blocks

=== test\_exceed\_memory ===

Error: Requested size (2097152 bytes) exceeds maximum block size (1024 bytes)

Error: Requested size (2097152 bytes) exceeds memory pool size

McKusick usage factor: 0

Pow2 usage factor: 0

=== test\_invalid\_deallocation ===

Error: Attempt to deallocate invalid pointer

McKusick: allocated & deallocated 10 blocks

Pow2: allocated & deallocated 10 blocks

**Вывод:** В результате выполнения проекта будет получено понимание работы различных аллокаторов памяти, их сильных и слабых сторон. Сравнительный анализ позволит сделать выводы о том, какой из алгоритмов более эффективен в зависимости от условий использования и требований к производительности. McKusick Allocator лучше для сценариев с частыми небольшими запросами памяти. Pow2 Allocator лучше для больших блоков, но приводит к большему количеству фрагментации.