ЛАБОРАТОРНАЯ РАБОТА №3

Введение

Виртуальная память является ключевым механизмом современных операционных систем, позволяющим абстрагировать физическую память от процессов. Она позволяет каждому процессу иметь своё собственное непрерывное адресное пространство и считать, что это пространство используется только им, не задумываясь об использовании физической памяти другими процессами. Операционная система достигает этого разбивая адресное пространство на страницы фиксированного размера (например, 4 КБ) и управляет их отображением в физическую память с помощью таблиц страниц.

Таблица страниц — это структура данных, которая сопоставляет виртуальные адреса с физическими. Каждый процесс имеет свою таблицу страниц, которая хранит соответствие виртуального адреса (номер страницы) с физическим адресом (номер страницы в физической памяти). Механизм управления памятью (MMU, Memory Management Unit) при помощи этих таблиц преобразует виртуальные адреса в физические во время выполнения программы. Кроме непосредственно изоляции, такая организация памяти позволяет реализовать ряд дополнительных механизмов, повышающих эффективность выполнения программы. Одним из таких механизмов является, например, механизм свопинга (swapping). В случае нехватки оперативной памяти система может использовать доступное дисковое пространство - вытеснять страницы на диск. В последствии если требуемая виртуальная страница отсутствует в физической памяти происходит так называемое прерывание отсутствия страницы (page fault), при котором операционная система загружает страницу из дисковой подкачки (swap) обратно в оперативную память.

Таким образом, виртуальная память предоставляет каждому процессу иллюзию большого непрерывного адресного пространства, при этом повышается безопасность и эффективность использования физической памяти.

Задание

Целью данной лабораторной работы является изучение базовых принципов работы виртуальной памяти. Необходимо реализовать простейший алгоритм на языке C.

Реализация должна включать имплементацию 4 функций:

- `int vm_alloc(int worker_id)` — выделение виртуальной страницы для процесса.
- `void vm_free(int worker_id, int page_no)` — освобождение ранее выделенной страницы.

- `void vm_write(int worker_id, int page, int value)` — запись значения в виртуальную страницу.
- `int vm_read(int worker_id, int page)` — чтение значения из виртуальной страницы.

Заголовочный файл:

```
#ifndef STUDENT_H
#define STUDENT_H

#include <stdbool.h>



#define PHYSICAL_MEMORY_SIZE 1000
#define MAX_WORKERS 50
#define MAX_VIRTUAL_PAGES_PER_WORKER 10000

// physical memory
extern int real_memory[PHYSICAL_MEMORY_SIZE];

// your virtual memory interface
int vm_alloc(int worker_id);  // allocate page (return it's number or -1)
void vm_free(int worker_id, int page_no); // free page

void vm_write(int worker_id, int page, int value); // write some value to the page
int vm_read(int worker_id, int page); // read some value from the page
#endif
```

Воркеры - "процессы", запрашивающие память и используещие ее. Они любезно передают свой номер в функции управления виртуальной памятью (что, в реальности, конечно, так не происходит), которые необходимо реализовать. К лабораторной также прилагается код тестирования алгоритма. Его менять нельзя. После написания необходимо скомпилировать программу вместе с кодом тестирования и заголовочным файлом и убедиться, что тесты прошли.

Отчет о работе должен содержать код и описание реализации.