Virtual Storage Simulator

ОГЛАВЛЕНИЕ

ИСХОДНЫЕ ФАЙЛЫ

Список исходных файлов с описанием

— **sim.cpp**, **sim.hpp** — исходный код симулятора.

— **vm.cpp**, **vm.hpp** — исходный код библиотеки объектов, моделирующих поведение ВП.

— **config.hpp** — файл конфигурации, в котором описаны все константы, влияющие на поведение всей программы в целом.

ОПИСАНИЕ МОДЕЛЕЙ И ОБЪЕКТОВ

Моделируемые сущности

В этой программе моделируются адреса, но не данные, которые хранятся по этим адресам. Структуризация памяти страничная, размер страницы указывается в байтах в виде степеней двойки (например: 2048 байт будет записано как 11, так как $2^{11} = 2048$). Размеры адресных пространств в количестве страниц указаны в классе **Computer**. Само РАП моделируется как логический массив в классе **MemoryAddressSpace**.

Модель процесса включает в себя только лишь логику использования памяти (в том числе тот факт, что код процесс также занимает сколько-то страниц памяти), при этом логика задач, выполняемых этим процессом, в программе не рассматривается.

В этой программе ОС управляет виртуальной памятью, а значит и обеспечивает ТП. Для каждого процесса создаётся своя ТП, каждая запись в которой состоит из трёх элементов – виртуальный адрес из ВАП, моделируемого для каждого процесса в отдельности, реальный адрес из РАП, моделируемого для всей системы в целом, а также из атрибута для указания дополнительной служебной информации. Моделирование ВАП обеспечивается этими таблицами переадресации.

Модель процессора занимается только лишь тем (возможно пока), что выполняет задачу преобразования адреса.

Модель АС пока не прорабатывалась.

Описание классов

Класс **Computer** хранит в себе информацию о ключевых свойствах системы, таких как размер реальной памяти, архивной среды и страницы.

В нём будет реализован механизм расчёта загруженности системы и накладных расходов на страничный обмен.

Данные:

- **uint64_t rm_size** — размер реальной памяти в количестве страниц.
- **uint64_t ae_size** — размер архивной среды в количестве страниц.

— **uint64_t page_size** — размер страницы в байтах.

— **MemorySpaceAddress rm** — объект, хранящий в себе РАП.

Методы:

— Геттеры и сеттеры данных.

Класс **CPU** моделирует работу процессора, решая задачу преобразования виртуального адреса в реальный.

Данные отсутствуют.

Методы:

— **void Convert(PageNumber address).** В этом методе реализована задача преобразования адреса.

Класс **OS** моделирует работу ОС, в том числе решает задачи управления виртуальной памятью, обеспечивает таблицу переадресации каждому инициализированному процессу.

Данные:

— **TranslationTable *translation_tables[MAX_PROCESS_NUM]** — массив таблиц переадресации. Память под каждую из ТП выделяется динамически.

— **int current_translation_table_index** — текущий свободный индекс в массиве таблиц переадресации. Если существую две ТП, то этот индекс будет равен 2, так как индексы 0, 1 будут уже заняты.

Методы:

— **void CallCPU(bool WriteFlag)** – вызов процесса для решения задачи преобразования.

— **void HandleInterruption()** – обработка прерывания по отсутствию страницы.

— **void InitializeProcess(Process *_process)** – реализация загрузки кода процесса в память.

— Конструктор по умолчанию.

— Деструктор для освобождения динамически выделенной под ТП памяти.

Класс **AE** моделирует работу архивной среды.

Класс **Process** — базовая модель процесса.

Данные:

- **int translation_table_index** — индекс таблицы переадресации для этого процесса.
- **uint64_t memory_usage** – количество страниц, необходимое для размещения кода этого процесса в памяти.

Методы:

- Геттеры и сеттеры для данных.
- **void MemoryRequest(bool WriteFlag)** – метод, реализующий запрос памяти.
- **virtual void Work()** – метод, в котором описана некоторая работа, совершаемая этим процессом.

Описание моделей

Модель процесса **MyProcess** реализуется в одноимённом классе, унаследованном от класса **Process.** В методе **Work()** этой модели реализована много итерационная задача обращения матрицы большой размерности.