Лабораторная работа № 1

МОДЕЛИРОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ

ВИРТУАЛЬНОЙ ПАМЯТЬЮ

**1. ЦЕЛЬ РАБОТЫ**

Получить представление о механизме управления виртуальной памятью, закрепить навыки программирования на языке C(C++) с использованием динамических структур данных и прямого доступа к файлy.

**2. ОБЩИЕ СВЕДЕНИЯ**

Под виртуальной памятью понимают память, находящуюся вне адресного пространства задачи [2]. С помощью механизма виртуализации можно снять ограничения на объем непосредственно адресуемого пространства, естественно в пределах дополнительной памяти, в качестве которой часто используется внешняя память.

Одна из возможных реализаций виртуальной памяти основана на страничной организации, когда весь объем необходимой памяти разбивается на фрагменты-страницы, располагаемые во внешней памяти. Некоторая часть страниц постоянно находится в оперативной памяти. Система управления по мере необходимости производит замещение (подкачку) недостающих страниц.

Страницы, располагаемые в оперативной памяти, снабжаются набором признаков (атрибутов), позволяющих эффективно управлять процессом замещения. В число атрибутов могут входить:

‑ флаг модификации (признак записи данных на страницу);

‑ флаг защиты от записи (признак запрета/разрешения записи);

‑ счетчик обращений к странице;

‑ счетчик времени пребывания страницы в памяти и т.п.

**3. ПОСТАНОВКА ЗАДАЧИ И РЕКОМЕНДАЦИИ**

**ПО ВЫПОЛНЕНИЮ**

Разработать пакет программ для управления виртуальной памятью. Разработке подлежат следующие процедуры (функции) :

‑ процедура инициализации виртуального массива vini;

‑ процедура записи в массив vput;

‑ процедура чтения из массива vget;

‑ процедура вычисления адреса элемента addres.

Рекомендуемые прототипы процедур :

VIRTUAL \*vini(long size, int Type),  
где VIRTUAL ‑ структура дескриптора виртуального массива;

size ‑ размер инициализируемого виртуального массива в эле-  
 ментах;

Type ‑ размер элемента в байтах.

Алгоритм:

‑ выравнивает размер массива на границу страницы;

‑ создает файл для размещения виртуального массива. Размер файла определяется общим числом страниц в массиве;

‑ создает и инициализирует дескриптор виртуального массива;

‑ возвращает адрес дескриптора или NULL, если нет возможности создать массив.

int vput(VIRTUAL \*array, long index, VTYPE\*value),

где array ‑ адрес дескриптора используемого виртуального массива;  
 index ‑ индекс элемента массива;  
 value ‑ адрес значения, записываемого в массив;  
 VTYPE ‑ тип массива(должны поддерживаться все базовые типы   
 языка).

Алгоритм:

‑ вызывает функцию addres для получения адреса элемента;

‑ записывает значение по этому адресу;

‑ возвращает результат завершения операции.

int vget(VIRTUAL \*array, long index, VTYPE\*value),

где array ‑ адрес дескриптора используемого виртуального массива;  
 index ‑ индекс элемента массива;  
 value ‑ адрес переменной, получающей значение;  
 VTYPE ‑ тип массива (должны поддерживаться все базовые типы   
 языка).

Алгоритм:

‑ вызывает функцию addres для получения адреса элемента;

‑ копирует значение с этого адреса в указанную переменную;

‑ возвращает результат завершения операции.

void \*addres(VIRTUAL \*array, long index),

где аргументы определяются как указано выше.

Алгоритм:

‑ определяет номер страницы, на которой находится требуемый элемент;

‑ проверяет наличие страницы в памяти;

‑ при отсутствии страницы в памяти случайным образом выбирает страницу для замещения;

‑ проверяет статус выбранной страницы (флаг модификации);

‑ если страница модифицирована, выгружает ее в файл;

‑ если необходимо, загружает в память новую страницу;

‑ вычисляет относительный адрес элемента на странице;

‑ возвращает абсолютный адрес элемента или NULL, если обнаружена ошибка.

Рекомендуемая структура дескриптора:

struct VIRTUAL {  
 FILE \*Fp; /\*файловый указатель виртуального  
 массива\*/

int Type; /\*размер элемента в байтах\*/

int Status[NPAGES]; /\*статус страницы (флаг модифика-  
 ции)\*/   
 int Number[NPAGES]; /\*номер страницы\*/   
 char Page[NPAGES\*PAGESIZE]; /\*буфер страниц\*/  
 }   
 NPAGES ‑ количество страниц, присутствующих в памяти (рекомен-  
 дуется 2);  
 PAGESIZE ‑ размер страницы (рекомендуется 512 байт).

Ошибочные ситуации :

‑ недостаточно места на диске для размещения файла;

‑ ошибка файловой операции;

‑ недостаточно оперативной памяти для размещения дескриптора;

‑ индекс элемента выходит за границы массива;

Тестирующая программа:

‑ генерирует набор тестовых данных, размещенных случайным образом по массиву (тип данных произвольный, может задаваться преподавателем при защите) (см. описание функций vput и vget);

‑ выводит на экран набор тестовых данных до записи в массив и после чтения из массива.

‑ при завершении не уничтожает файлы виртуального массива.

Эти файлы уничтожаются вручную после просмотра дампа файла при защите.

Время, отводимое на выполнение , ‑ 8 -10 ч .

Представление результатов лабораторной работы :

‑ демонстрация тестирующей программы и дампа файла виртуального массива;

‑ отчет по лабораторной работе установленного образца. Отчет должен включать описание (можно графическое) всех структур, используемых в системе управления виртуальной памятью.

**4. КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Что такое виртуальная память?

2. Чем ограничивается объем виртуальной памяти?

3. Стратегии замещения страниц.

4. Как определить номер страницы по индексу элемента?

5. Как определить адрес страницы в файле?

6. Как определить смещение элемента внутри страницы?