Операционные системы

Лекция № 8

Память и адресное пространство процесса

(4 часа)

Основные определения

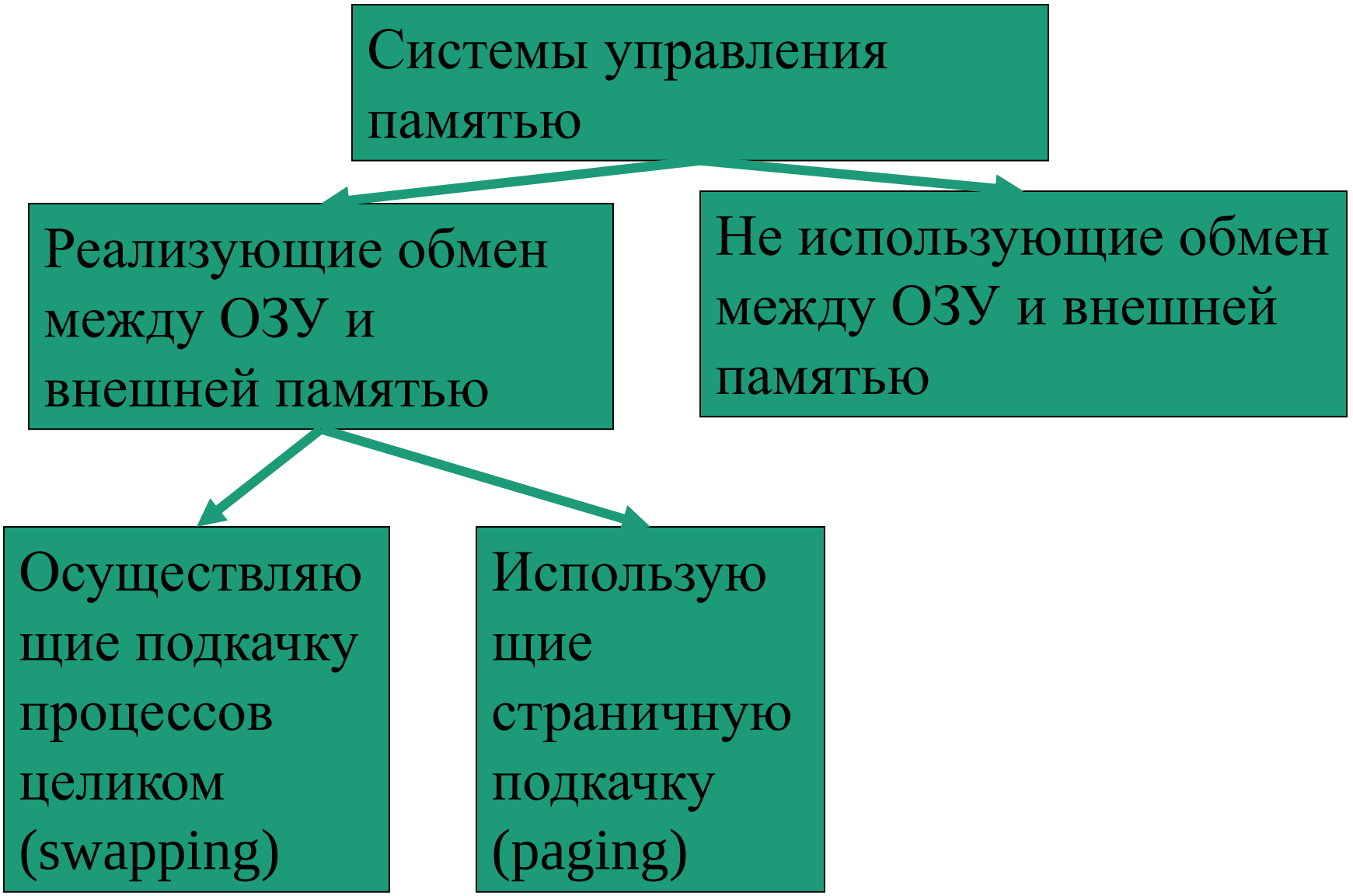
Управлением памятью (**memory management**) называется задача распределения памяти операционной системой для размещения в ней нескольких процессов.

Часть операционной системы, отвечающая за управление памятью, называется **модулем управления памятью** или **менеджером памяти**.

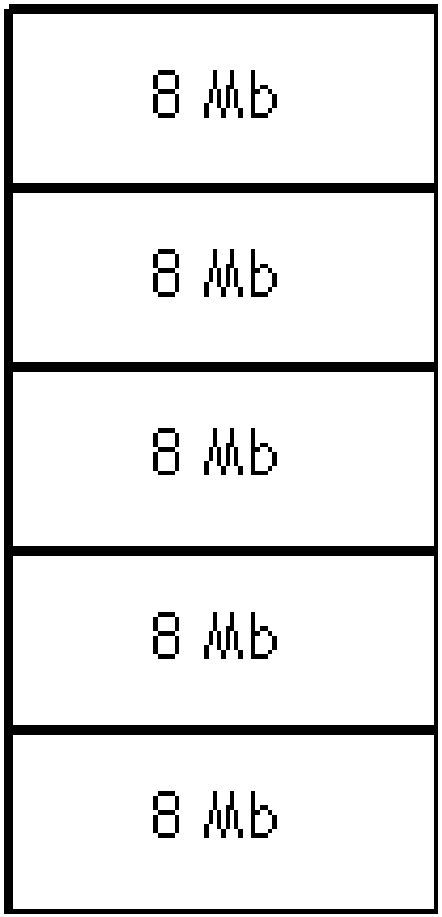
Требования к стратегиям управления памятью

- **Перемещение** (relocation) – перемещение программы из одной области памяти в другую;
- **Защита** (protection) – защита памяти одного процесса от других процессов;
- **Совместное использование** (sharing) – возможность обращаться к одной области памяти нескольким процессам;
- **Логическая организация** (logical organization) – совмещение линейной организации памяти и модульности ПО;
- **Физическая организация** (physical organization) - организация взаимодействия между первичной и вторичной памятью.

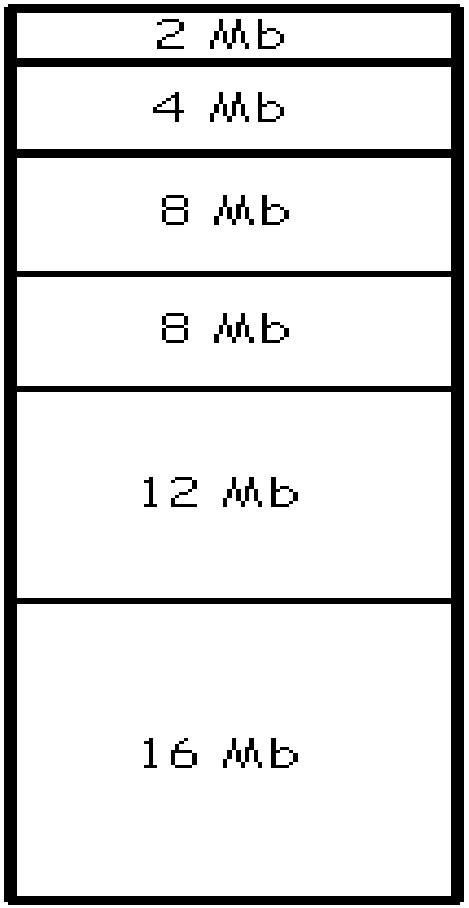
Основное управление памятью



Фиксированное распределение памяти



Фиксированное распределение памяти



Алгоритм размещения

- Простейший - каждый процесс размещается в наименьшем разделе, способном уместить данный процесс
- Одна очередь для всех разделов

Достоинства

- Простота
- Минимум требовании к ОС
- Накладные расходы процессов невелики

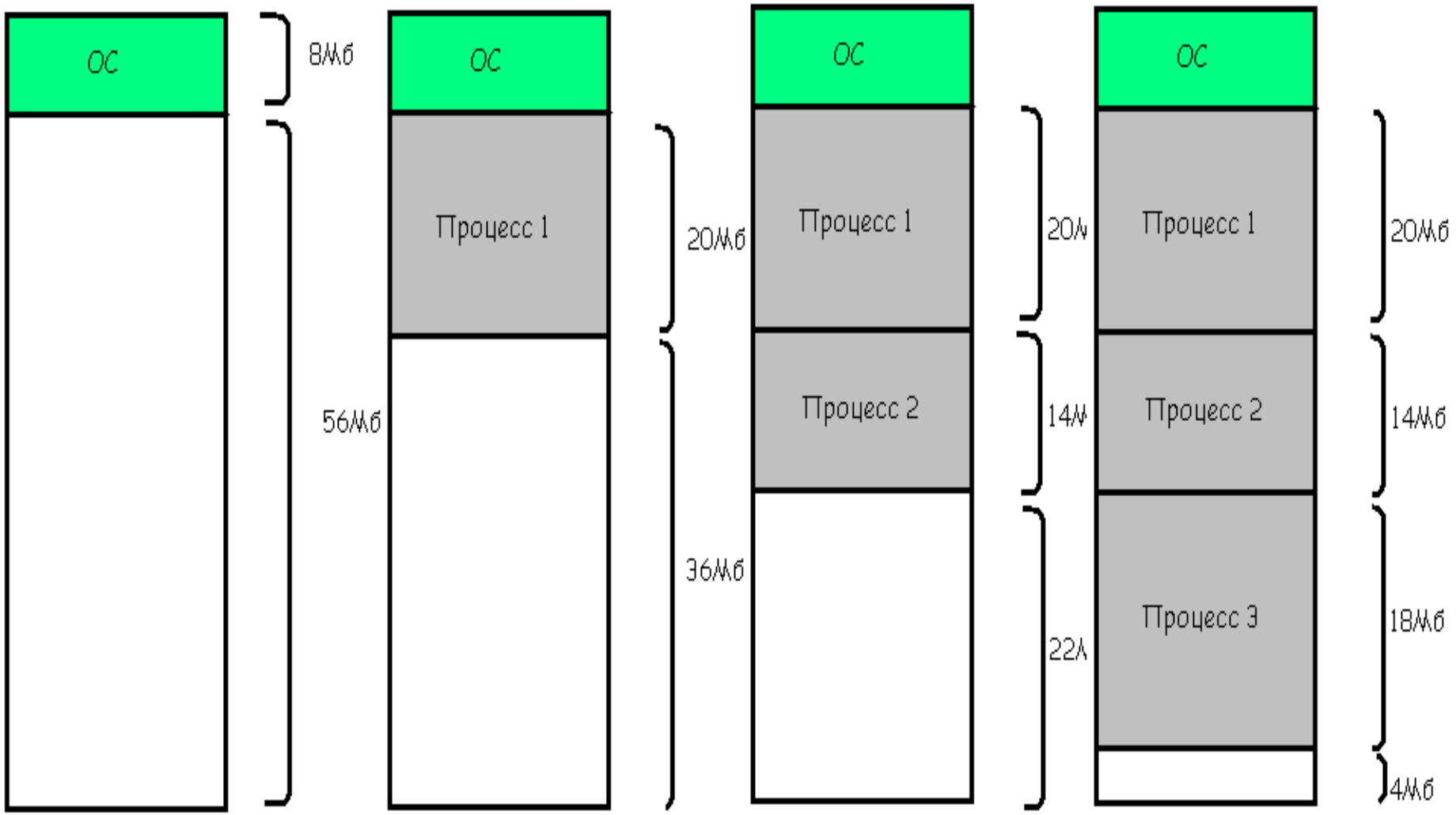
Недостатки

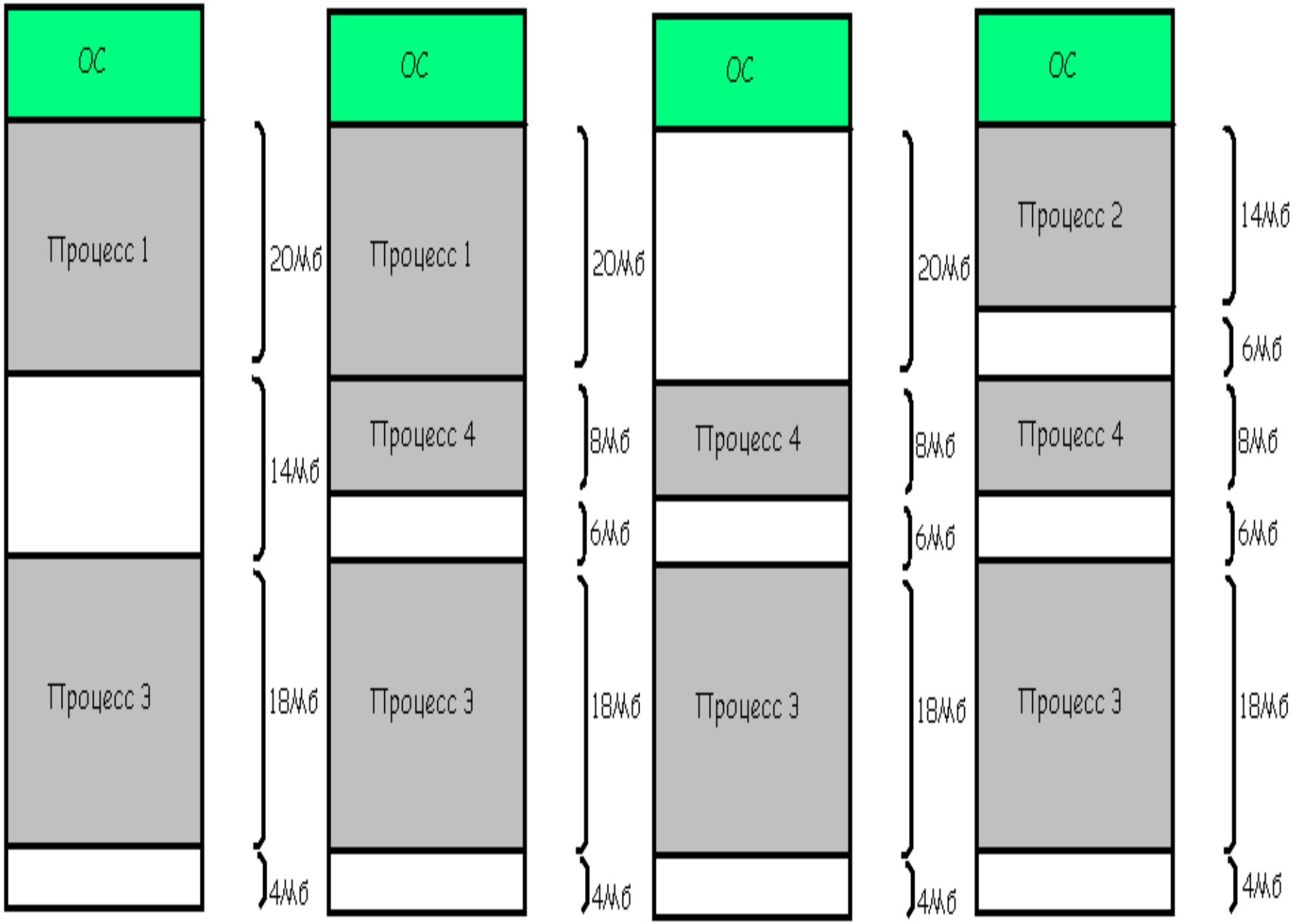
- Ограниченное количество активных процессов
- Программа может быть слишком велика для загрузки в память => использование оверлеев
- Небольшие процессы неэффективно используют память (внутренняя фрагментация)

Основные понятия

Ситуация появления неиспользуемой памяти в результате того, что загружаемый блок по размеру меньше раздела, называется **внутренней фрагментацией** (internal fragmentation).

Динамическое распределение разделов




ОС
Процесс 1
Процесс 3
ОС
20Мб
14Мб
18Мб
4Мб
Процесс 1
Процесс 4
Процесс 3
ОС
20Мб
8Мб
6Мб
18Мб
4Мб
Процесс 4
Процесс 3
ОС
20Мб
8Мб
6Мб
18Мб
4Мб
Процесс 2
Процесс 4
Процесс 3
14Мб
6Мб
8Мб
6Мб
18Мб
4Мб

Основные определения

Явление увеличения фрагментации в результате которого блок памяти меньше, чем загружаемая программа называется **внешней фрагментацией** (external fragmentation).

*Сильно фрагментированной становится память, **внешняя** по отношению ко всем разделам (в отличие от внутренней фрагментации).*

Уплотнение памяти

Периодически системы с динамическим распределением разделов требуют проведения **уплотнения памяти** (memory compaction) – перемещения процессов в памяти в смежные блоки и сбор свободной памяти в один блок.

Алгоритм размещения

- Наилучший подходящий
- Первый подходящий
- Следующий подходящий

Достоинства

- Отсутствие внутренней фрагментации
- Более эффективное использование основной памяти, чем в случае фиксированного распределения

Недостатки

- Неэффективное использование CPU из-за необходимости уплотнения для противодействия внешней фрагментации

Типы адресов

- Логический адрес – ссылка на ячейку памяти, не зависящая от текущего расположения данных в памяти.
- Перед тем как получить доступ к этой ячейке памяти, необходимо **транслировать** логический адрес в физический.

Типы адресов

- Относительный адрес –частный случай логического адреса: адрес определяется положением относительно некоторой известной точки (обычно – начала программы).

Типы адресов

- Физический адрес (абсолютный) – действительное расположение интересующей нас ячейки памяти

Простая страничная организация

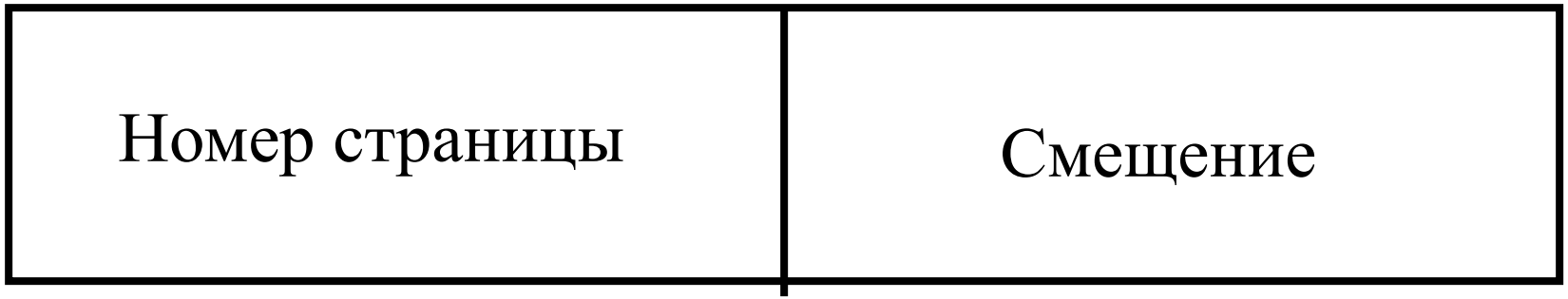
- Основная память разделена на одинаковые блоки относительно небольшого фиксированного размера – **кадры** (frames), или **фреймы.**
- Процесс разделен на одинаковые блоки, известные как **страницы** (pages) той же длины, что и кадры.
- Все страницы загружаются в доступные кадры, не обязательно последовательные.

Основные определения

Для каждого процесса поддерживается специальная таблица:

Таблица страниц – системная структура, задающая расположение кадров каждой страницы процесса.

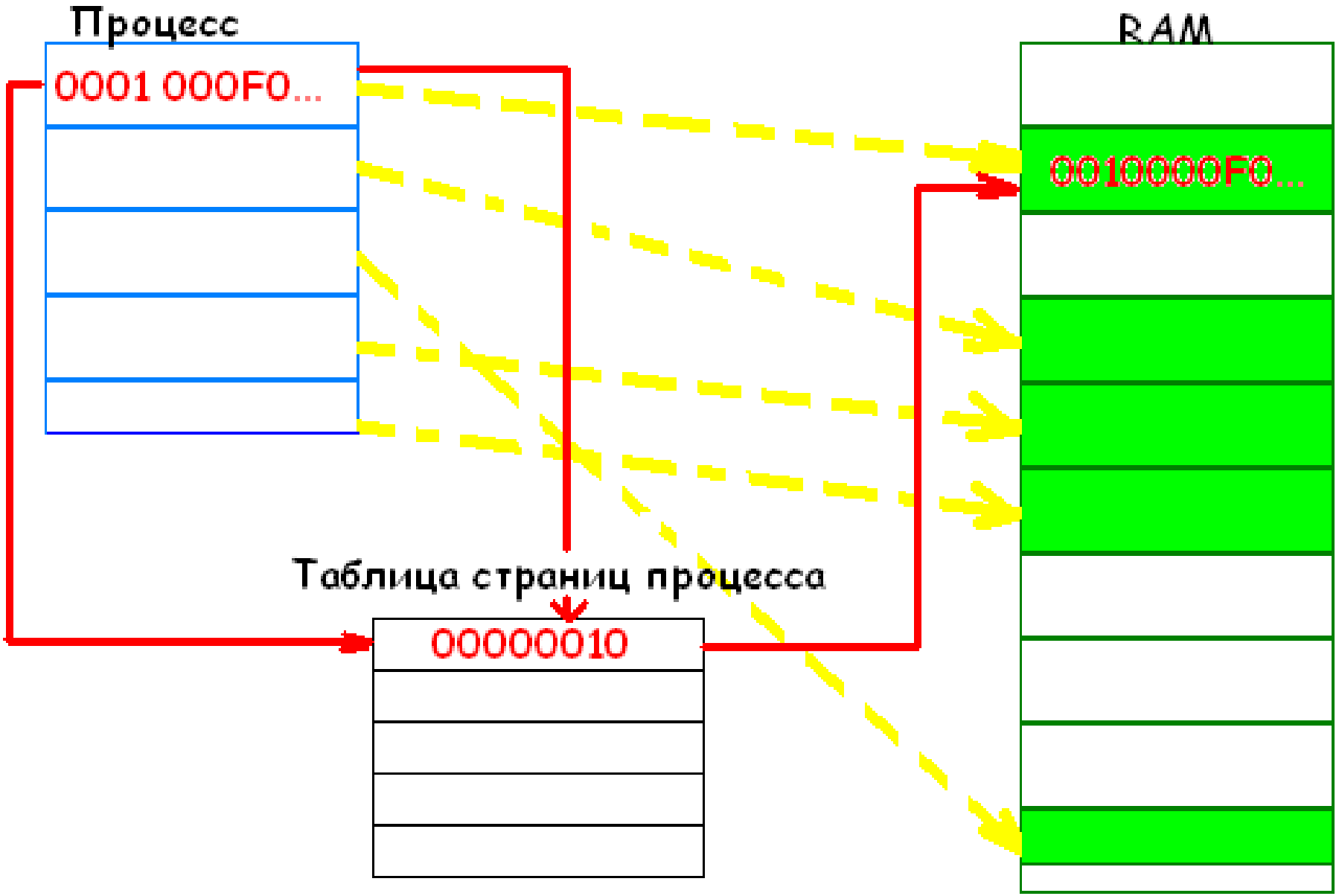
Структура логического адреса



Трансляция адресов

- Выделить из логического адреса n битов слева, получив номер страницы
- Используя номер страницы в качестве индекса в таблице страниц процесса, найти номер кадра k
- Начальный физический адрес кадра – k x 2^m; физический адрес – полученное число плюс смещение. Такой адрес не надо вычислять – он получается в результате простого добавления номера кадра к смещению

Простая страничная организация



Простая страничная организация

Достоинства:

- Отсутствие внешней фрагментации

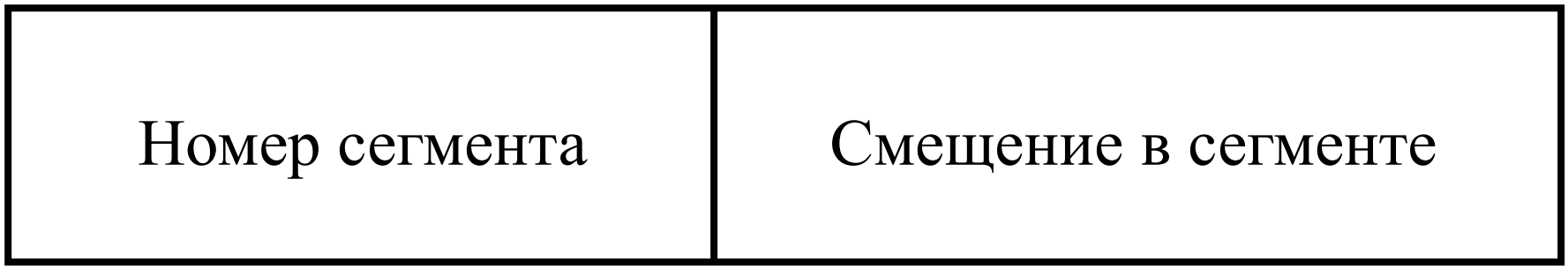
Недостатки:

- Наличие небольшой внутренней фрагментации.

Простая сегментация

- Программа и связанные с ней данные разделяются на ряд **сегментов**
- Процесс загружается путем загрузки всех сегментов в динамические, необязательно смежные, разделы.

Структура логического адреса

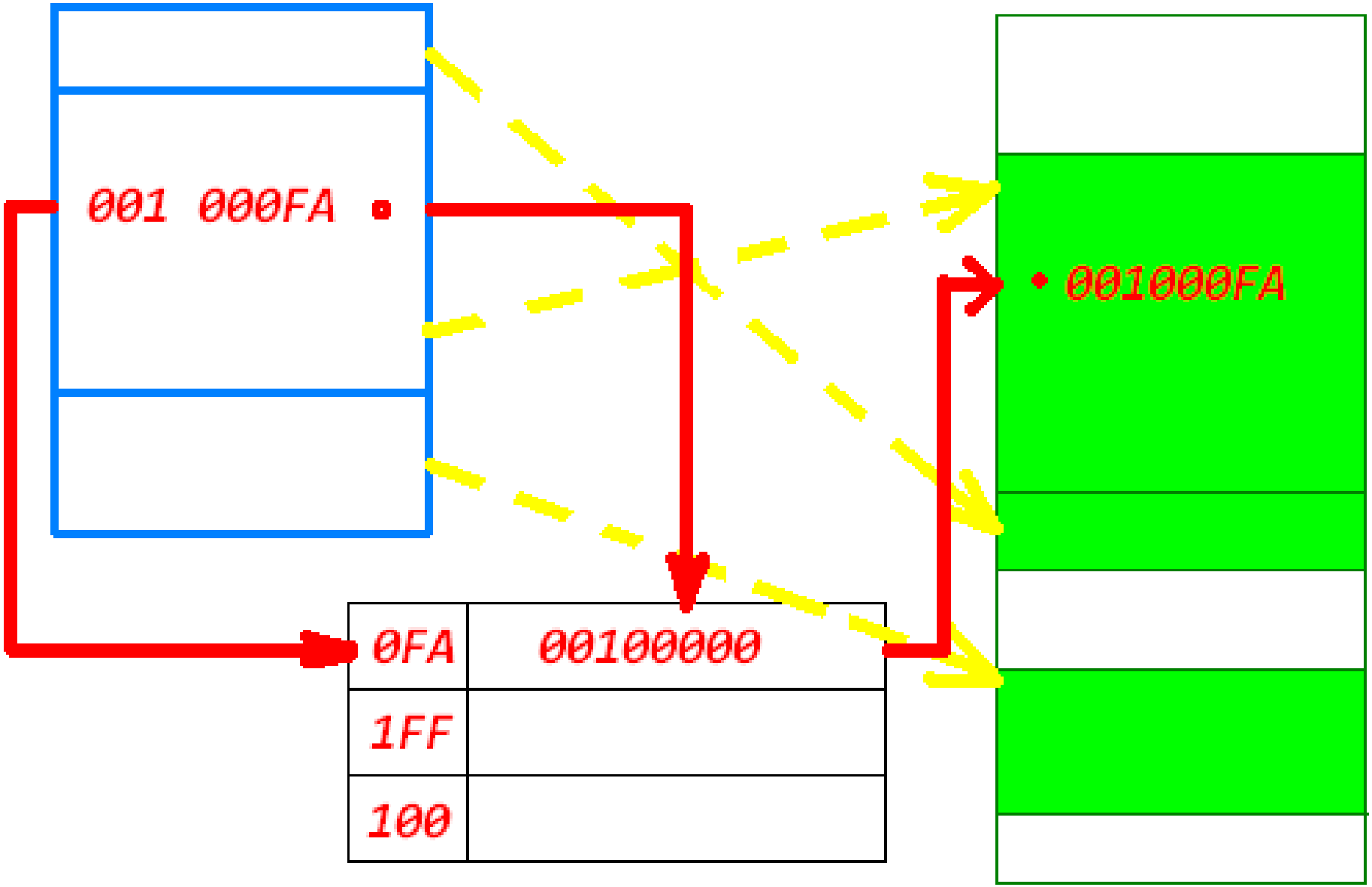


Для разрешения логического адреса используется **таблица сегментов процесса**. Таблица сегментов, в отличие от таблицы страниц, содержит не только адрес начала сегмента, но и его длину.

Трансляция адресов

- Выделить из логического адреса n битов слева, получив номер сегмента.
- Используя номер сегмента в качестве индекса в таблице сегментов процесса, найти физический адрес начала сегмента
- Сравнить смещение (крайние справа m бит) с длиной сегмента. Если смещение больше длины сегмента – адрес некорректен
- Требуемый физический адрес представляет собой сумму физического адреса начала сегмента и смещения

Простая сегментация



Виртуальная память

Так как:

- Все обращения к памяти - логические адреса;
- Логические адреса транслируются в физические динамически;
- Процесс разбит на ряд частей, не обязательно располагающихся непрерывным блоком

то наличие всех страниц или сегментов процесса в основной памяти одновременно не является обязательно!

Основные определения

- Поскольку процесс выполняется только в основной памяти, эта память называется также **реальной** (real memory).
- Вся доступная процессу память, включая вытесненную во вторичную память, называется **виртуальной** (virtual memory).

Виртуальная память

Исполнение процесса начинается с загрузки одного или нескольких блоков процесса в ОЗУ. При обращений к отсутствующему блоку CPU генерирует прерывание ошибки доступа к памяти, блокирует процесс и загружает нужный блок в основную память.

Ненужные блоки могут быть вытеснены во внешнюю память – в файлы страниц (page file) или в файлы подкачки (swap file).

Виртуальная память

Эффективность подхода гарантирует доказанный эмпирически **ПРИНЦИП ЛОКАЛИЗАЦИИ**:

Обращения к коду и данным в процессе имеют тенденцию к кластеризации: в течение небольшого времени для работы требуется только небольшая часть процесса.

Часть процесса, располагающаяся в данный момент в основной памяти называется **резидентным множеством процесса** или **рабочим множеством**.

Особенности реализации управления памятью

- Вид реализации: страничная адресация, сегментация или обе технологии;
- Стратегия выборки;
- Стратегия замещения;
- Управление резидентным множеством;
- Стратегия очистки.

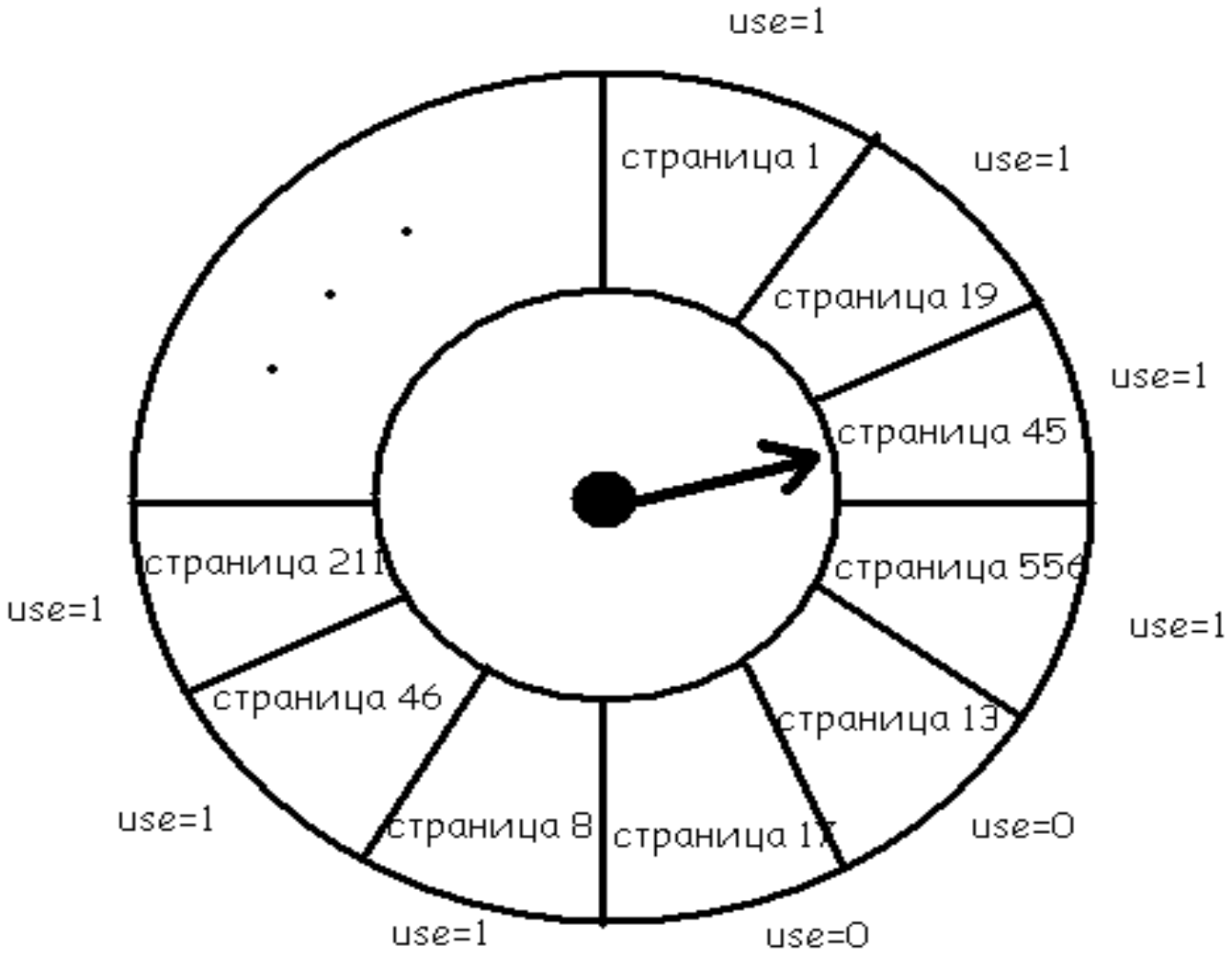
Стратегия выборки

- Выборка по требованию – страница передается в основную память только при обращении к ней;
- Предварительная выборка – загружается не только страница, вызвавшая прерывание, но и несколько соседних страниц

Основные алгоритмы выбора замещаемой страницы

- Оптимальный алгоритм
- FIFO (first in first out);
- LRU (least recently used) – вытесняется дольше всех неиспользовавшаяся;
- LFU (least frequently used) – вытесняется наименее часто используемая;
- NUR (not used recently), аналог LRU – вытесняются страницы на которые дольше всех не было записи;

Часовая стратегия (clock policy)



Вариации: учитывая бит модификации

- использован давно, не модифицирован *(u* = 0, *m* = 0);
- использован недавно, не модифицирован *(u* = 1, *m* = 0);
- использован давно, модифицирован *(u* = 0, *m* = 1);
- использован недавно, модифицирован *(u* — 1, *m* = 1).

Вариации: учитывая бит модификации

1. Сканируем буфер, начиная с текущего положения. В процессе сканирования бит использования не изменяется. Первый блок с состоянием *(u* = 0, *m* = 0) замещается
2. Если выполнение первого шага алгоритма не увенчалось успехом, ищем блок с параметрами *{u* = 0, *m* = 1). Если найден – замещается. В процессе выполнения данного шага у всех просмотренных блоков сбрасывается бит использования
3. Повторяем шаги 1 и 2.

Блокировка кадров

Некоторые кадры памяти могут быть **заблокированы**, т.е. страница, хранящаяся в этом кадре в настоящее время не может быть замещена. (ядро ОС, основные управляющие структуры)

Управление резидентным множеством

Размер резидентного множества:

- Фиксированный;
- Переменный;

Область видимости замещения:

- Глобальная;
- Локальная;

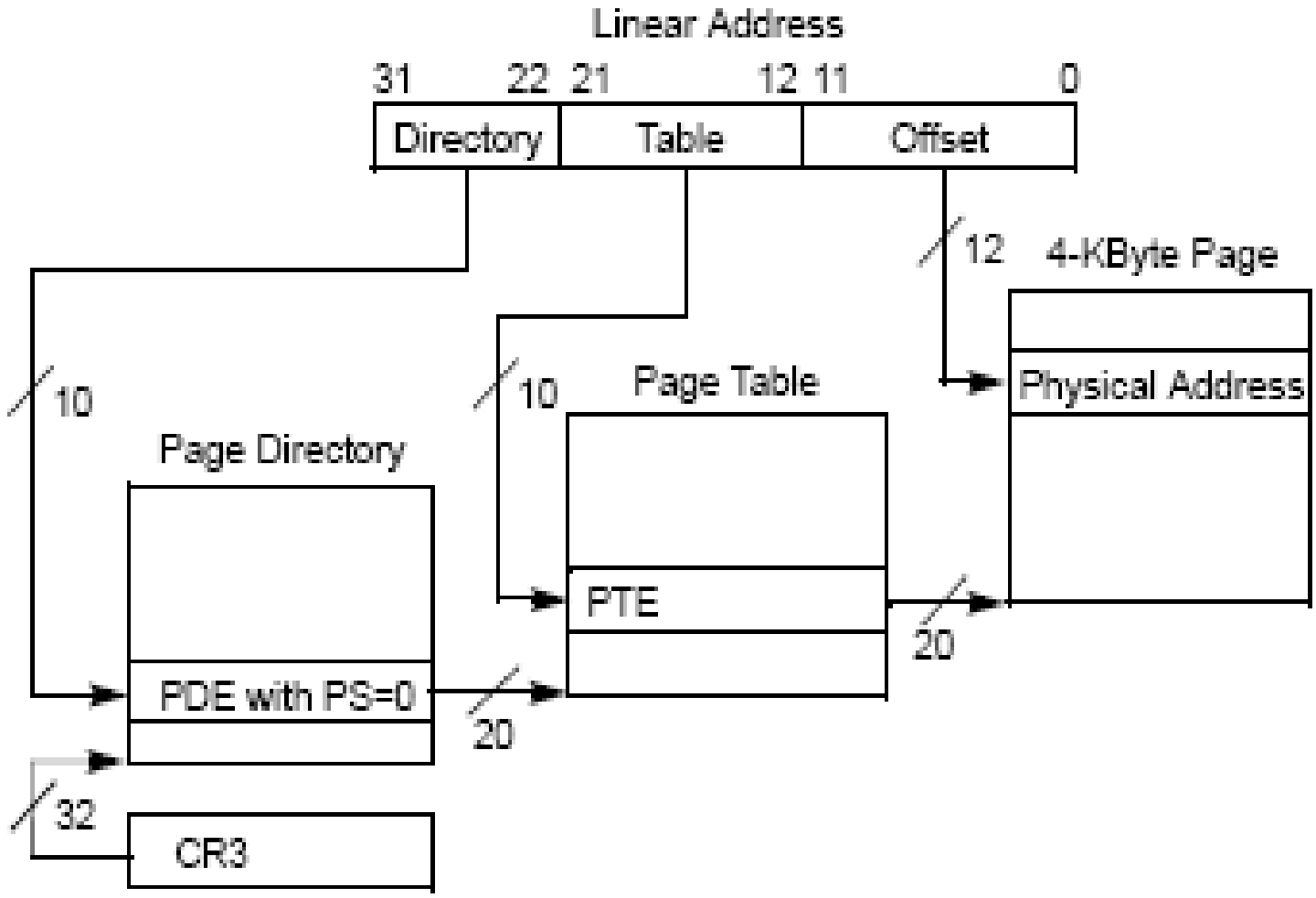
Стратегия очистки

- По требованию;
- Предварительная;

Страничная организация виртуальной памяти

- Все страницы процесса делятся на резидентные и нерезидентные (не загруженные в основную память в данный момент);
- Необходимые нерезидентные страницы автоматически подгружаются;
- Поддерживаются страницы размером 4Кб, 2Мб, 4Мб, 1Гб (для различных архитектур CPU и различных режимов) и от 1 до 4-х уровневые структуры для таблиц страниц

Вид таблиц страниц



Таблицы страниц

Некоторые флаги PTE:

- P (Present) - присутствие страницы в основной памяти;
- R/W – разрешение на чтение/запись;
- U/S – владелец страницы;
- PCD – кэширование страницы отключено;
- A (Accessed) – была операция чтения из страницы;
- D (Dirty) – в страницу была произведена запись;

Страничная организация виртуальной памяти

Достоинства:

- Нет внешней фрагментации;
- Более высокая степень многозадачности;
- Большое виртуальное адресное пространство;
- Процесс может быть больше доступной основной памяти.

Недостатки:

- Накладные расходы из-за сложной системы управления памятью

Сегментация виртуальной памяти

- Не требуется одновременно загружать все сегменты процесса;
- Необходимые нерезидентные сегменты автоматически загружаются.

Сегментация виртуальной памяти

Достоинства:

- Нет внутренней фрагментации;
- Более высокая степень многозадачности;
- Большое виртуальное адресное пространство;
- Поддержка защиты и совместного использования.

Недостатки:

- Накладные расходы из-за сложной системы управления памятью

Виртуальная память в Windows

Используется страничная организация виртуальной памяти.

Управлением памятью занимается менеджер виртуальной памяти VMM.

Состояния страниц реальной памяти:

Valid – используется процессом;

Modified – записывается на диск;

Standby – удаляется из рабочего множества;

Free – освобождена, но не обнулена;

Zeroed – обнулена, может использоваться;

Bad – в нерабочем состоянии.

Виртуальная память в Windows

На основании флагов PTE каждой странице назначается доступ:

- PAGE_NOACCESS – доступ запрещен;
- PAGE_READONLY – доступ только на чтение;
- PAGE_READWRITE – доступ на чтение и запись в страницу;
- PAGE_EXECUTE – разрешено выполнение;
- PAGE_GUARD – исключение.

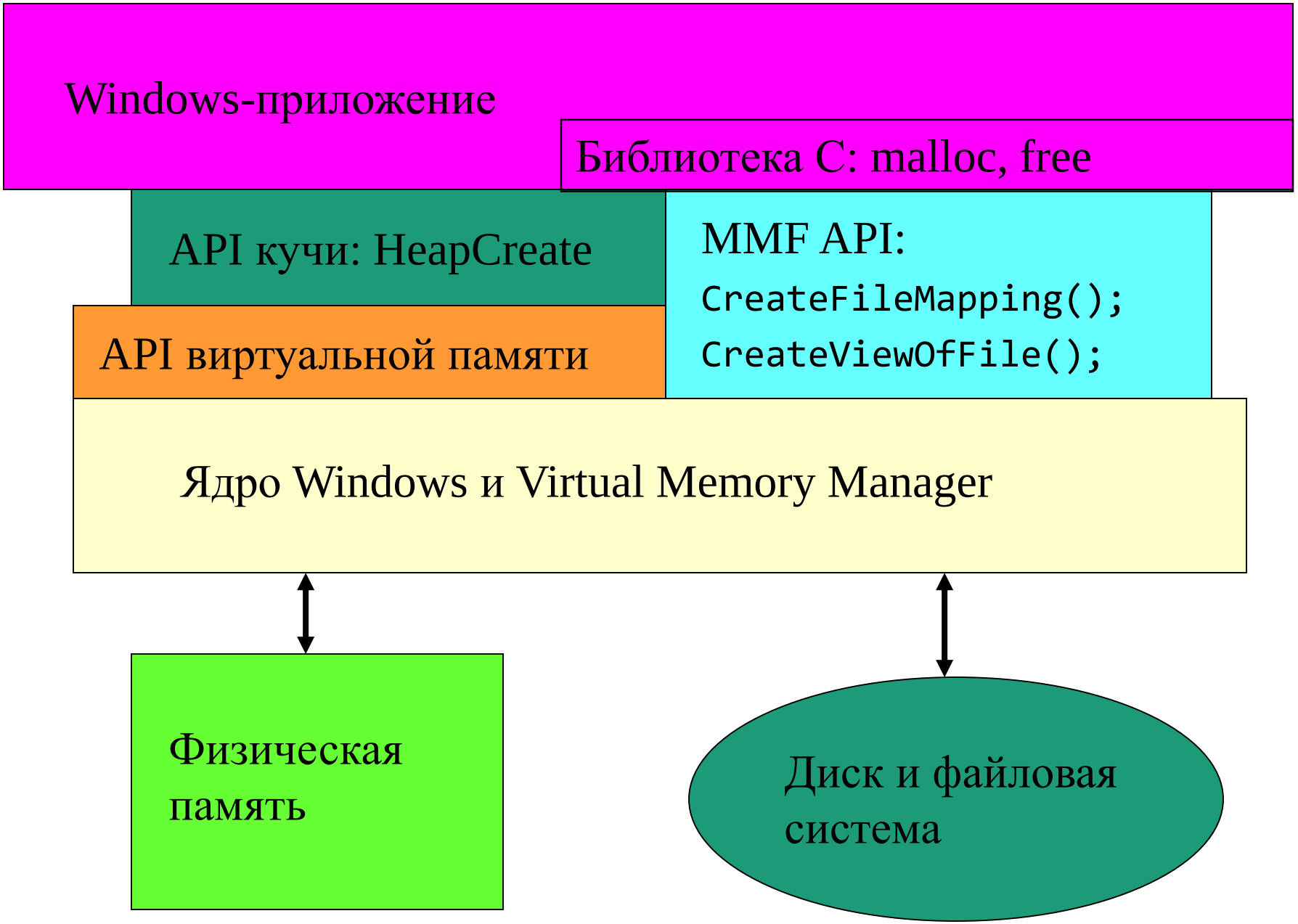
При замещении страниц используется алгоритм LRU локально.

Виртуальная память в Windows

Виртуальное адресное пространство процесса - 4Гб делится на 2 части: 2:2 или 3:1.

Страницы виртуальной памяти могут быть:

- Свободны (free);
- Распределены для использования (committed);
- Зарезервированы процессом, но не использоваться (reserved);


Windows-приложение
Библиотека C: malloc, free
API кучи: HeapCreate
MMF API:
CreateFileMapping();
CreateViewOfFile();
API виртуальной памяти
Ядро Windows и Virtual Memory Manager
Физическая память
Диск и файловая система

Работа с виртуальной памятью

Для резервирования и распределения:

```
LPVOID VirtualAlloc(
  LPVOID lpAddress,
  SIZE_T dwSize,
  DWORD flAllocationType,
  DWORD flProtect
);
```

`lpAddress` – адрес области для распределения или резервирования (или NULL);

`dwSize` – размер области;

`flAllocationType` – тип распределения;

`flProtect` – тип защиты доступа

Возвращается – адрес области, иначе – NULL.

В случае успеха память инициализируется нулями

Работа с виртуальной памятью

`lpAddress` и `dwSize` – выравниваются с учетом гранулярности выделения памяти в Windows;

`flAllocationType` – комбинация из `MEM_COMMIT`, `MEM_RESERVE`.

Работа с виртуальной памятью

Освобождение виртуальной памяти:

```
BOOL VirtualFree(
  LPVOID lpAddress,
  SIZE_T dwSize,
  DWORD dwFreeType
);
```

`lpAddress` – адрес освобождаемой области;

`dwSize` – размер области;

`dwFreeType` – тип операции (комбинация `MEM_DECOMMIT`, `MEM_RELEASE`).

При `MEM_RELEASE` `dwSize` должно быть равно 0.

Работа с виртуальной памятью

Для блокирования страниц виртуальной памяти:

```
BOOL VirtualLock(LPVOID lpAddress,  SIZE_T
  dwSize);
```

Для разблокирования:

```
BOOL VirtualUnlock(LPVOID lpAddress, SIZE_T
  dwSize);
```

Изменить атрибуты доступа к области виртуальной памяти:

```
BOOL VirtualProtect(LPVOID lpAddress,
SIZE_T dwSize,  DWORD flNewProtect,
PDWORD lpflOldProtect);
```

Отображение файлов в память

Ряд ОС позволяет отображать файлы в адресное пространство процесса.

Этот механизм используется для загрузки .exe и .DLL, создания разделяемой памяти и т.д.

Отображение файлов в память

- Отпадает необходимость выполнять операции непосредственного чтения и записи файлов;
- Становятся доступными для использования с файлами алгоритмы, ориентированные на работу в памяти (сортировки, обработка строк и т.д.);
- Увеличивается эффективность обработки файлов;
- Исчезает необходимость в прямом управлении буферами для чтения/записи файла и т.д.

Отображение файлов в память

Linux:

`mmap` – отображает файл в виртуальное адресное пространство процесса;

`munmap` – отсоединяет файл от виртуального адресного пространства процесса;

`msync` – синхронизирует содержимое памяти в виртуальном адресном пространстве процесса с содержимым файла на диске

Отображение файлов в память

```
#include <sys/mman.h>
#include <sys/types.h>


caddr_t mmap(caddr_t addr, int size,
  int prot, int flags, int fd, off_t
  pos);
int munmap(caddr_t addr, int size);
int msync(caddr_t addr, int size, int
  flags);
```

Отображение файлов в память

addr – адрес в виртуальном адресном пространстве процесса, начиная с которого отображается файл (если =0, ядро само выбирает адрес);

fd – дескриптор отображаемого файла полученный от open;

size – размер отображаемых данных из файла (должен быть <= размера файла);

prot – права доступа к отображению (PROT_READ, PROT_WRITE, PROT_EXEC)

flags – опции отображения (MAP_SHARED, MAP_PRIVATE, MAP_FIXED);

pos – начальная позиция в файле, с которой отображаются данные (0 или кратно размеру страницы);

Отображение файлов в память

Для `msync` параметр `flags` может принимать значения:

`MS_SYNC` - записать отображенные данные в файл на диске и дождаться окончания записи;

`MS_ASYNC` – то же, но вернуть управления сразу;

`MS_INVALIDATE` – отбросить содержимое отображенной области и перечитать данные с диска

Алгоритм работы

1. Откройте файл с нужными правами доступа (CreateFile);
2. Создайте объект ядра отображение файла в память с помощью CreateFileMapping;
3. Отобразите файл или его часть в адресное пространство процесса с помощью MapViewOfFile;
4. Выполните нужные действия с проекцией файла;
5. Отмените отображение файла UnmapViewOfFile;
6. Закройте объект ядра для отображения файла;
7. Закройте файл.

Создание объекта отображения файла в память

```
HANDLE CreateFileMapping(
HANDLE hFile,
LPSECURITY_ATTRIBUTES lpAttributes,
DWORD flProtect,
DWORD dwMaximumSizeHigh,
DWORD dwMaximumSizeLow,
LPCTSTR lpName);
```

Создает объект отображения файла и возвращает его дескриптор, иначе – NULL.

Создание объекта отображения файла в память

`hFile` – дескриптор файла;

`lpAttributes` – атрибуты защиты;

`flProtect` – права доступа к памяти (`PAGE_READONLY` и т.д.);

`dwMaximumSizeHigh, dwMaximumSizeLow` – старшая и младшая части размера объекта отображения;

`lpName` – имя объекта отображения.

Создание представления объекта отображения файла

```
LPVOID MapViewOfFile(
HANDLE hMapObject,
DWORD dwAccess,
DWORD dwOffsetHigh,
DWORD dwOffsetLow,
SIZE_T cbMap
);
```

Распределяет виртуальное адресное пространство процесса и проецирует на него файл с помощью объекта отображения. Возвращает указатель на распределенный блок или NULL.

Создание представления объекта отображения файла

`hMapObject` – дескриптор объекта отображения файла, возвращенный CreateFileMapping или OpenFileMapping;

`dwAccess` – защита страниц файла в памяти (FILE_MAP_WRITE, FILE_MAP_READ, FILE_MAP_ALL_ACCESS);

`dwOffsetHigh,dwOffsetLow` – старшая и младшая части смещения начала отображаемого участка от начала файла;

`cbMap` – размер отображаемого участка файла в байтах.

Освобождение представления объекта отображения файла

```
BOOL UnmapViewOfFile(LPVOID lpBaseAddress);
```

Освобождает память, выделенную представлению файла.

`lpBaseAddress` – должен быть равен адресу, полученному от `MapViewOfFile()`.