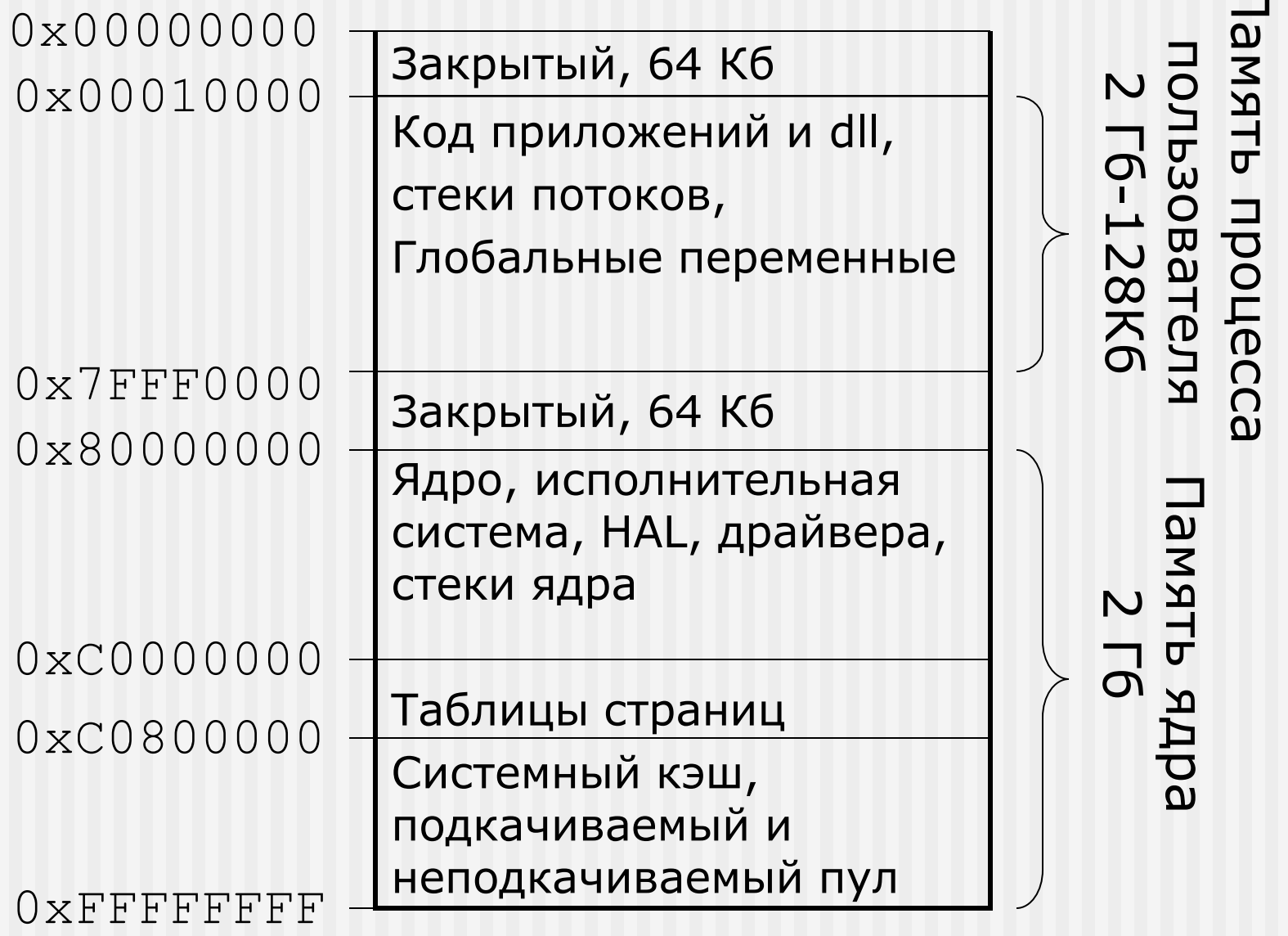
Оперативная память в Windows

Защита памяти в Windows

- Доступ к общесистемным структурам данных и пулам памяти, используемым системными компонентами режима ядра, возможен лишь из режима ядра — у потоков пользовательского режима нет доступа к соответствующим страницам.
- У каждого процесса имеется индивидуальное закрытое адресное пространство, защищенное от доступа потоков других процессов.
- Все процессоры, поддерживаемые Windows, предоставляют ту или иную форму аппаратной защиты памяти.
- Совместно используемые объекты имеют стандартные для Windows списки контроля доступа, проверяемые при попытках процессов открыть эти объекты.

Распределение памяти Windows



Выделение памяти процессу

- Гранулярность выделения: 64 Кб
- Двухступенчатая схема выделения:
  - Резервирование / Reserving
  - Выделение / Commiting
- Освобождение памяти:
  - Возврат / Decommiting
  - Освобождение / Releasing

Атрибуты защиты страниц

- PAGE_NOACCESS
  Доступ полностью запрещён
- PAGE_READONLY
  Только чтение
- PAGE_READWRITE
  Чтение и запись
- PAGE_EXECUTE
  Только выполнение кода
- PAGE_EXECUTE_READ
  Выполнение кода и чтение
- PAGE_EXECUTE_READWRITE
  Выполнение кода, чтение, запись
- PAGE_WRITECOPY
  Чтение, при записи предоставляется копия
- PAGE_EXECUTE_WRITECOPY
  Любые операции, при записи создаётся копия

Дополнительные флаги

- PAGE_NOCACHE
- PAGE_WRITECOMBINE
- PAGE_GUARD

Выделение памяти

```
LPVOID VirtualAlloc(
  LPVOID lpAddress, // region to reserve or commit
  SIZE_T dwSize,    // size of region
  DWORD flAllocationType,  // type of allocation
  DWORD flProtect    // type of access protection
);
```

flAllocationType:

- MEM_RESERVE
- MEM_COMMIT

flProtect:

- PAGE_READONLY
- PAGE_READWRITE
- …

Освобождение памяти

```
LPVOID VirtualFree(
  LPVOID lpAddress, // address of region
  SIZE_T dwSize,    // size of region
  DWORD dwFreeType,  // operation type
);
```

dwFreeType:

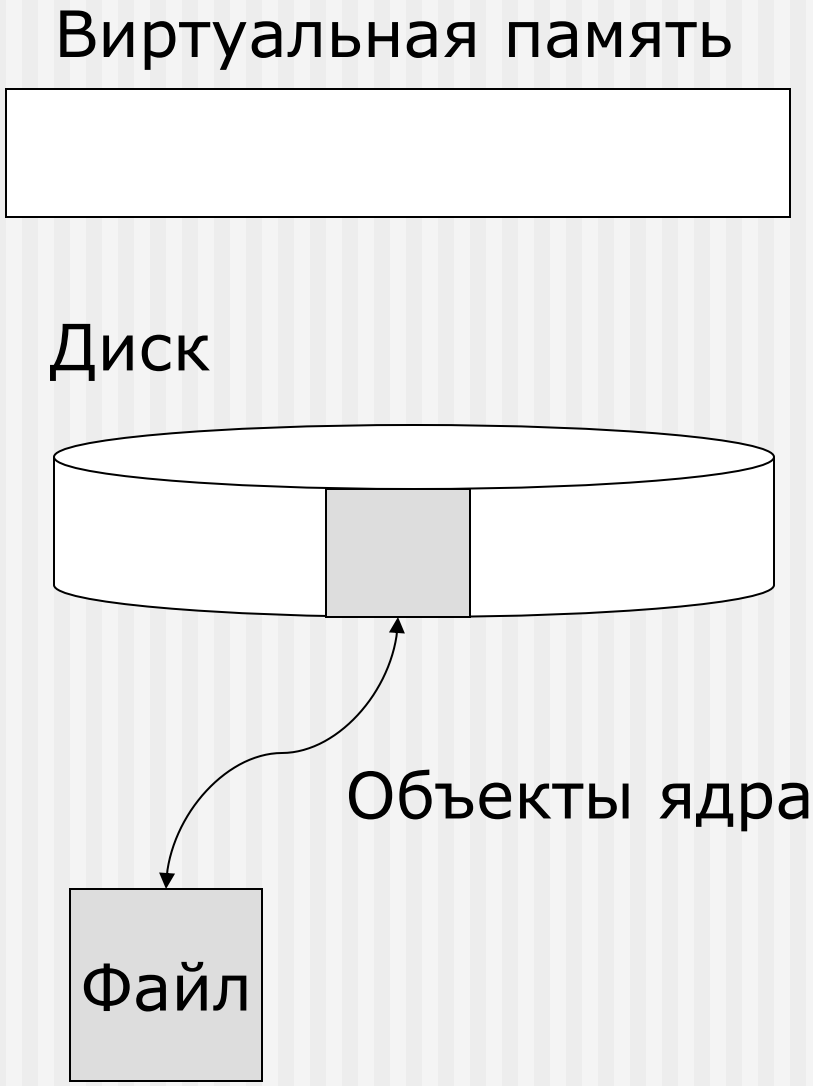
- MEM_RELEASE
- MEM_DECOMMIT

Проецируемые в память файлы

- Создать объект ядра файл
- Создать объект ядра проекция файла
- Проецирование части файла в адресное пространство

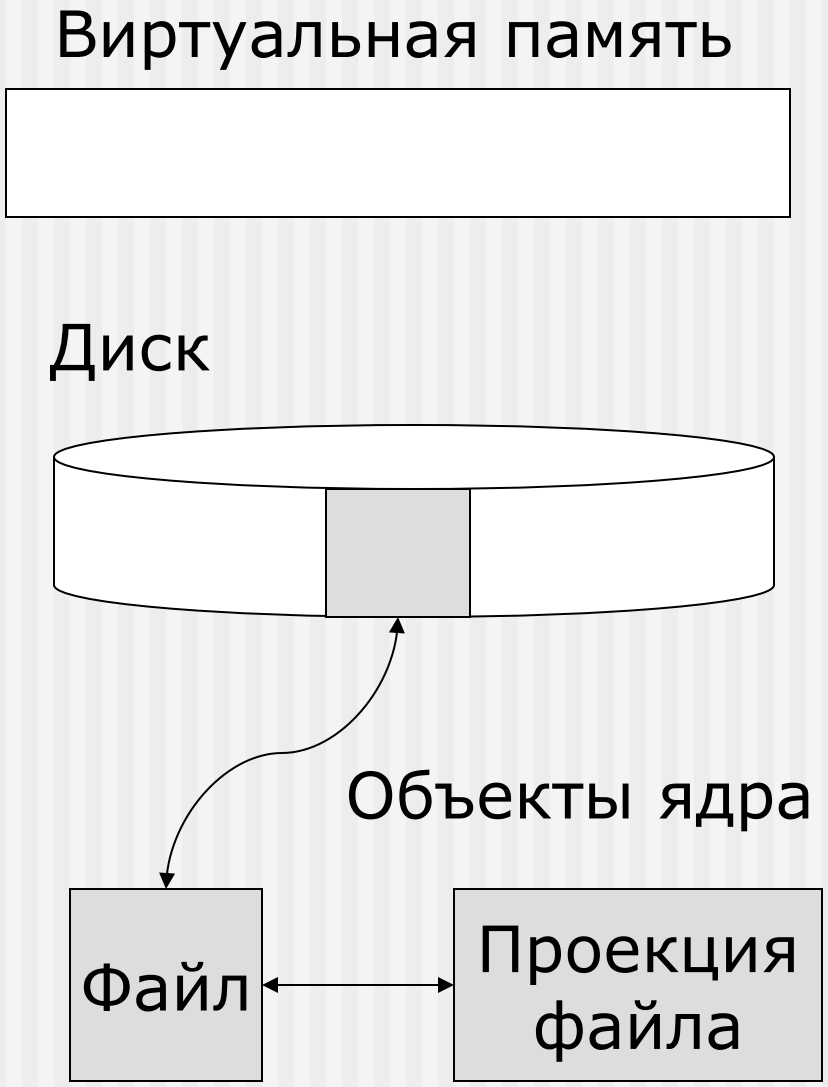
Проецируемые в память файлы

- **Создать объект ядра файл**
- Создать объект ядра проекция файла
- Проецирование части файла в адресное пространство



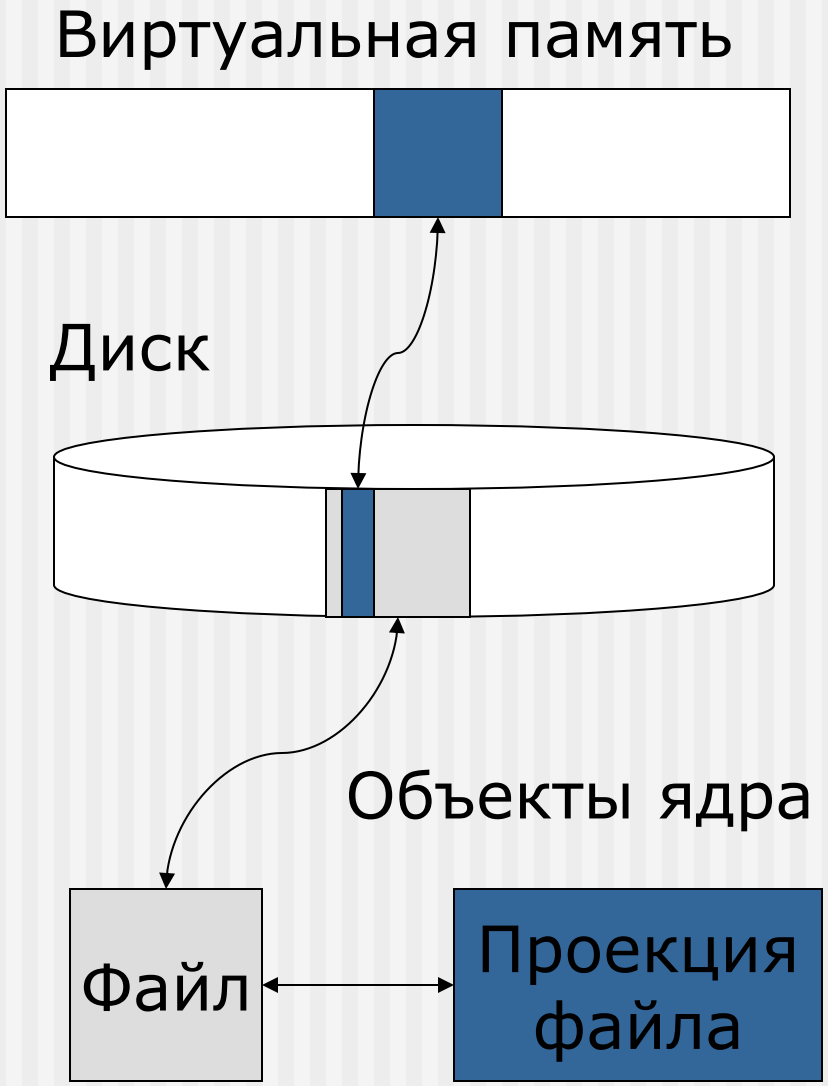
Проецируемые в память файлы

- **Создать объект ядра файл**
- **Создать объект ядра проекция файла**
- Проецирование части файла в адресное пространство



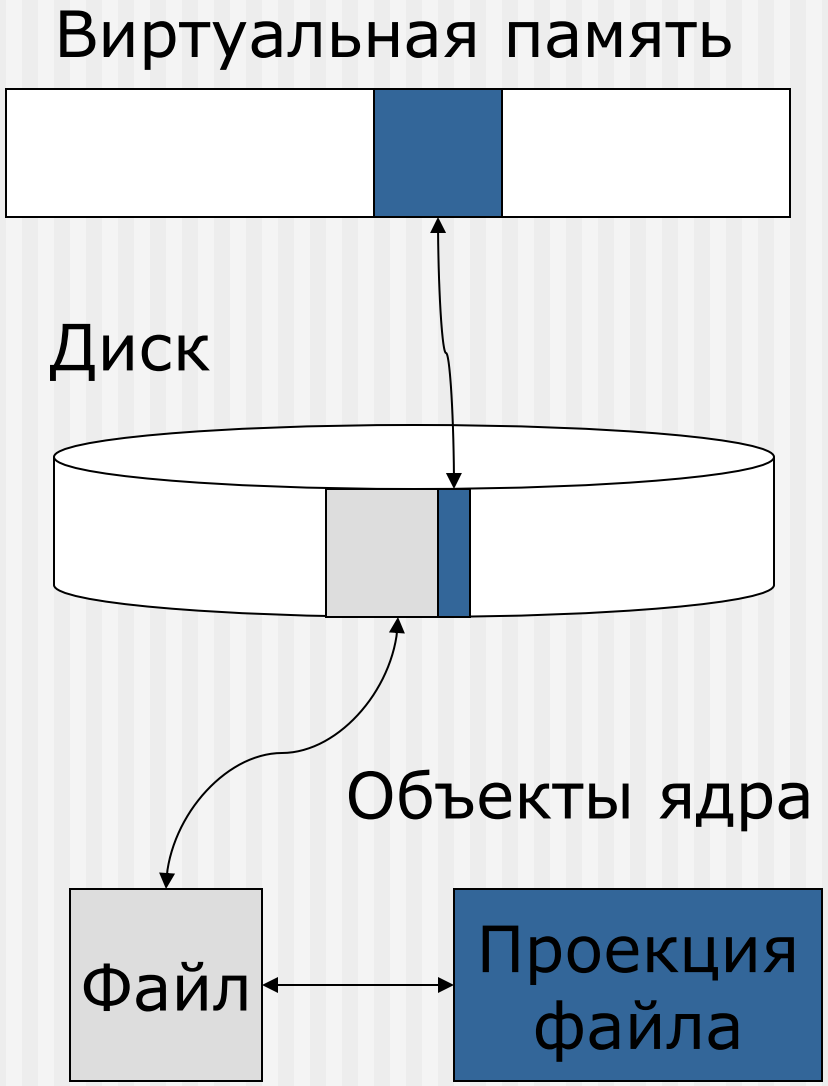
Проецируемые в память файлы

- **Создать объект ядра файл**
- **Создать объект ядра проекция файла**
- **Проецирование части файла в адресное пространство**



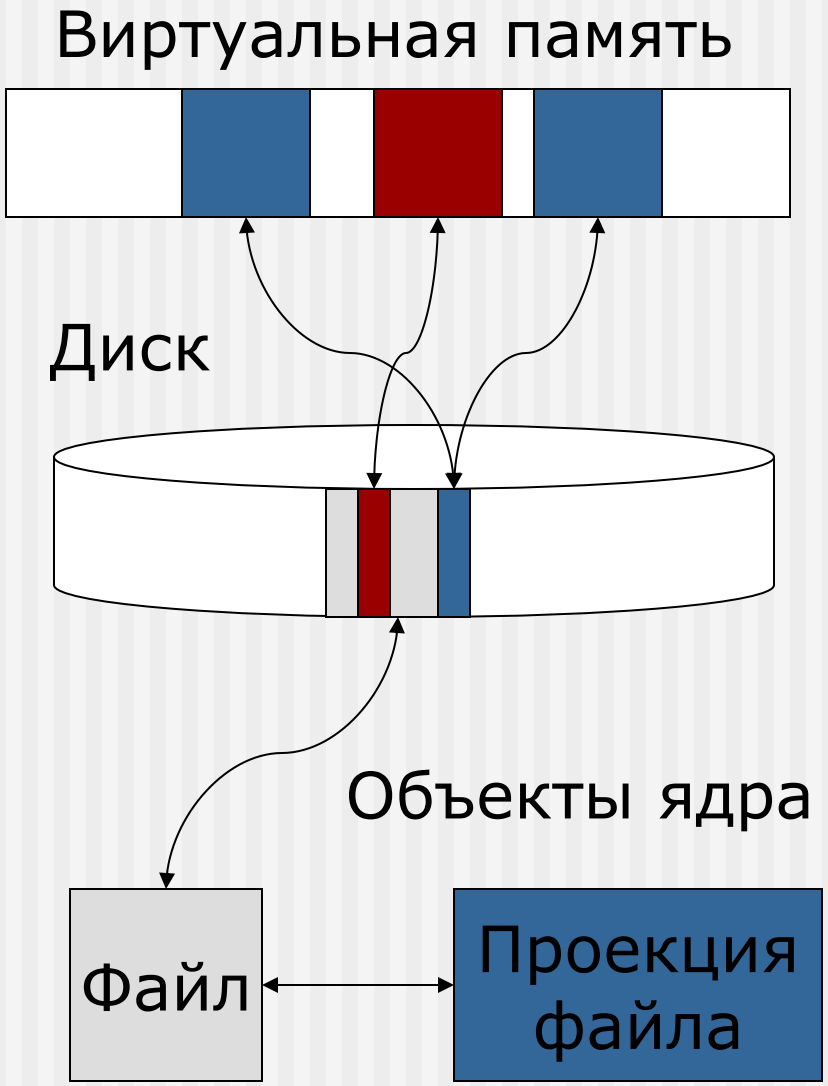
Проецируемые в память файлы

- **Создать объект ядра файл**
- **Создать объект ядра проекция файла**
- **Проецирование части файла в адресное пространство**
- **Изменение места проекции**



Проецируемые в память файлы

- **Создать объект ядра файл**
- **Создать объект ядра проекция файла**
- **Проецирование части файла в адресное пространство**
- **Изменение места проекции**



Создание объекта файл

```
HANDLE CreateFile(
 LPCTSTR lpFileName,          // file name
 DWORD dwDesiredAccess,       // access mode
 DWORD dwShareMode,           // share mode
 LPSECURITY_ATTRIBUTES lpSecurityAttributes,
 DWORD dwCreationDisposition, // how to create
 DWORD dwFlagsAndAttributes,  // file attributes
 HANDLE hTemplateFile  // handle to template file
);
```

Создание проекции файла

```
HANDLE CreateFileMapping(
HANDLE hFile,               // handle to file
LPSECURITY_ATTRIBUTES lpAttributes, // security
DWORD flProtect,            // protection
DWORD dwMaximumSizeHigh,//high-order DWORD of size
DWORD dwMaximumSizeLow, // low-order DWORD of size
LPCTSTR lpName              // object name
);
```

flProtect:

- PAGE_READONLY
- PAGE_READWRITE
- PAGE_WRITECOPY

Проецирование памяти

```
LPVOID MapViewOfFile(
HANDLE hFileMappingObject,
                    // handle to file-mapping object
DWORD dwDesiredAccess,      // access mode
DWORD dwFileOffsetHigh,
                    // high-order DWORD of offset
DWORD dwFileOffsetLow,
                    // low-order DWORD of offset
SIZE_T dwNumberOfBytesToMap
                    // number of bytes to map
);
```

dwDesiredAccess:

- FILE_MAP_READ
- FILE_MAP_WRITE
- FILE_MAP_ALL_ACCESS
- FILE_MAP_COPY

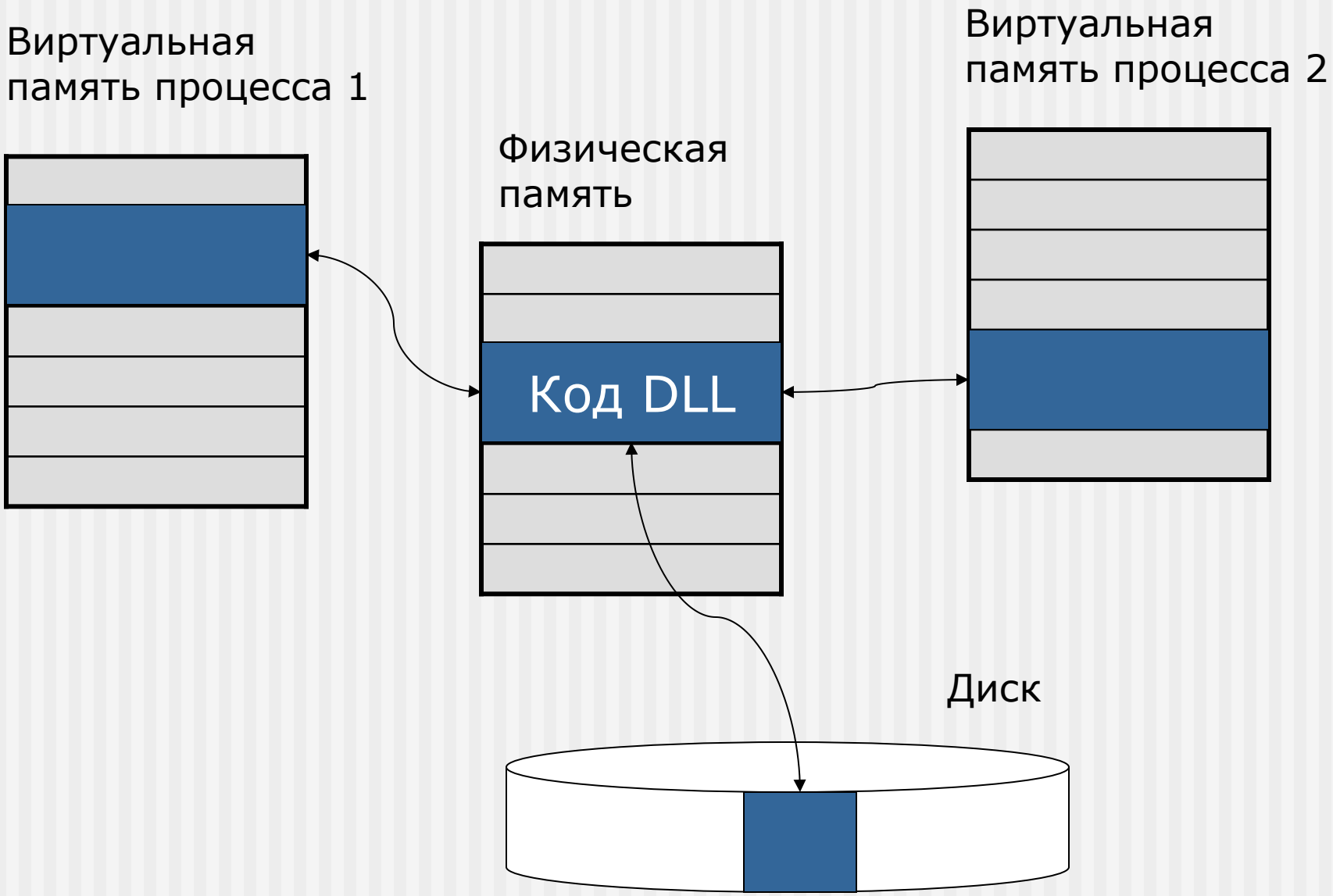
Другие функции

```
HANDLE OpenFileMapping(
  DWORD dwDesiredAccess,  // access mode
  BOOL bInheritHandle,    // inherit flag
  LPCTSTR lpName          // object name
);

BOOL UnmapViewOfFile(
LPCVOID lpBaseAddress   // starting address
);

BOOL FlushViewOfFile(
LPCVOID lpBaseAddress,  // starting address
SIZE_T dwNumberOfBytesToFlush
                        // number of bytes in range
);
```

Разделяемая память и проецируемые файлы



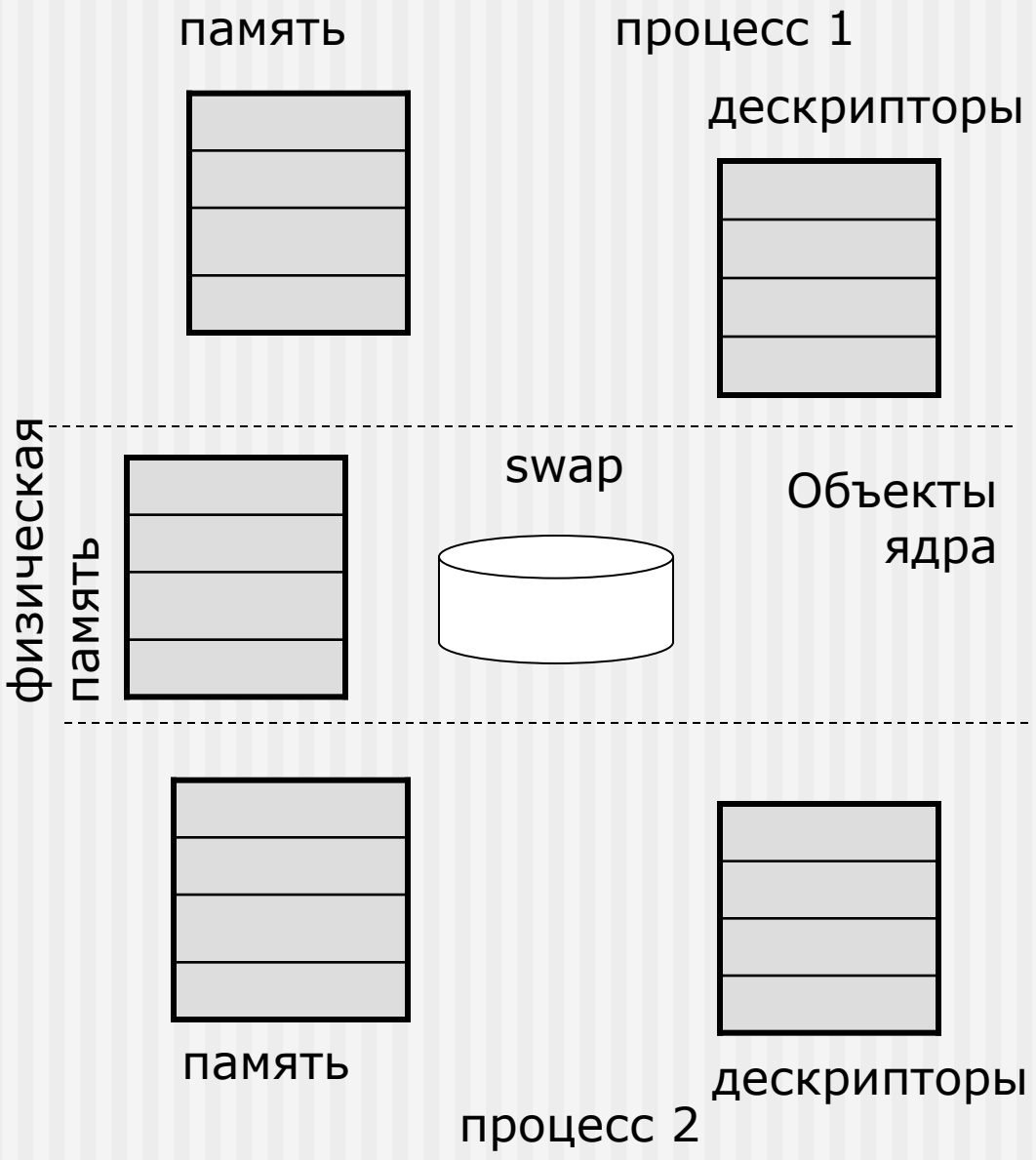
Создание разделяемой памяти

Процесс 1

- Создать объект проекция файла (hFile=NULL, задать размер и имя)
- Спроецировать память

Процесс 2

- Открыть объект проекция файла
- Спроецировать память



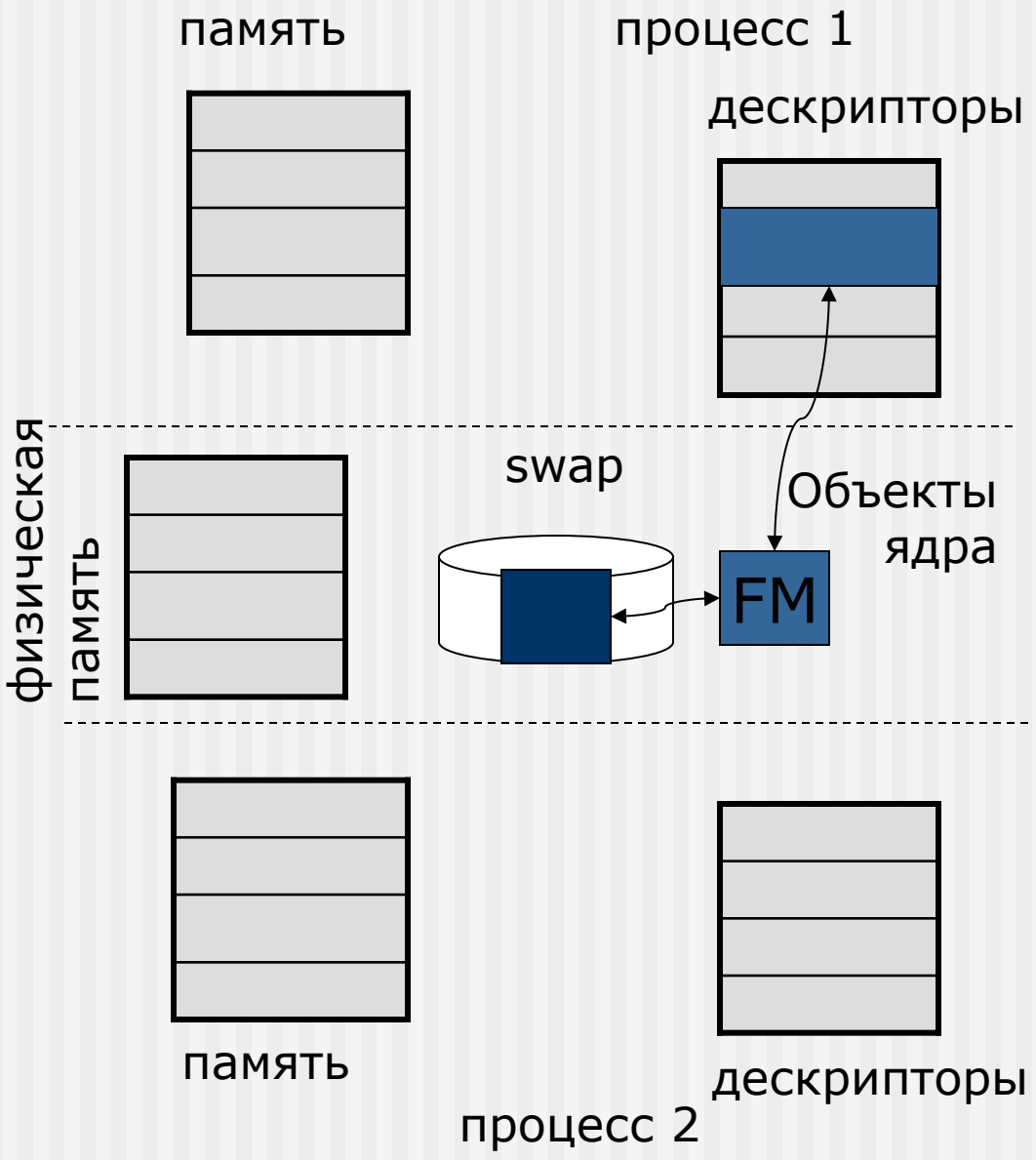
Создание разделяемой памяти

Процесс 1

- **Создать объект проекция файла (hFile=NULL, задать размер и имя)**
- Спроецировать память

Процесс 2

- Открыть объект проекция файла
- Спроецировать память



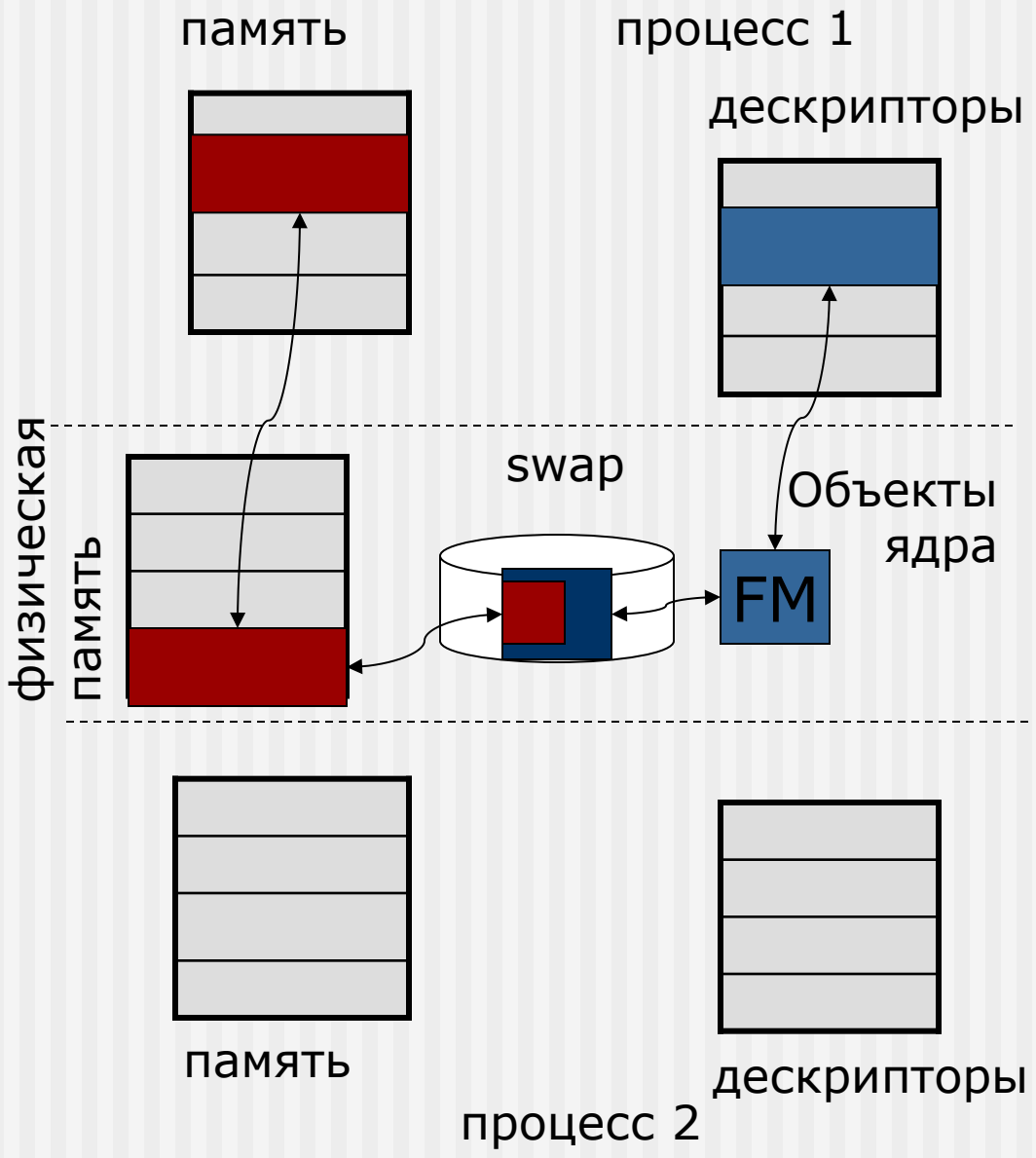
Создание разделяемой памяти

Процесс 1

- **Создать объект проекция файла (hFile=NULL, задать размер и имя)**
- **Спроецировать память**

Процесс 2

- Открыть объект проекция файла
- Спроецировать память



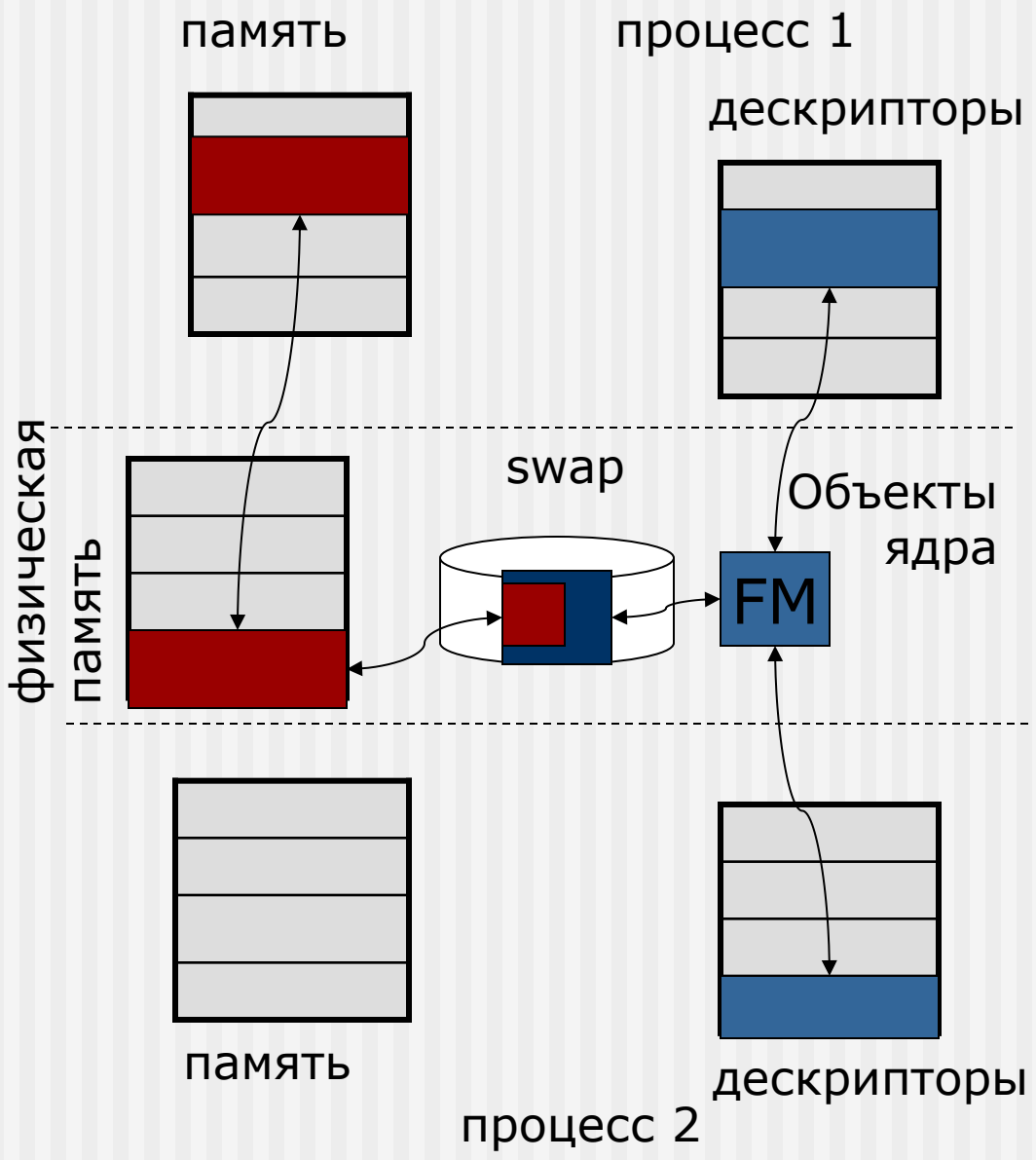
Создание разделяемой памяти

Процесс 1

- **Создать объект проекция файла (hFile=NULL, задать размер и имя)**
- **Спроецировать память**

Процесс 2

- **Открыть объект проекция файла**
- Спроецировать память



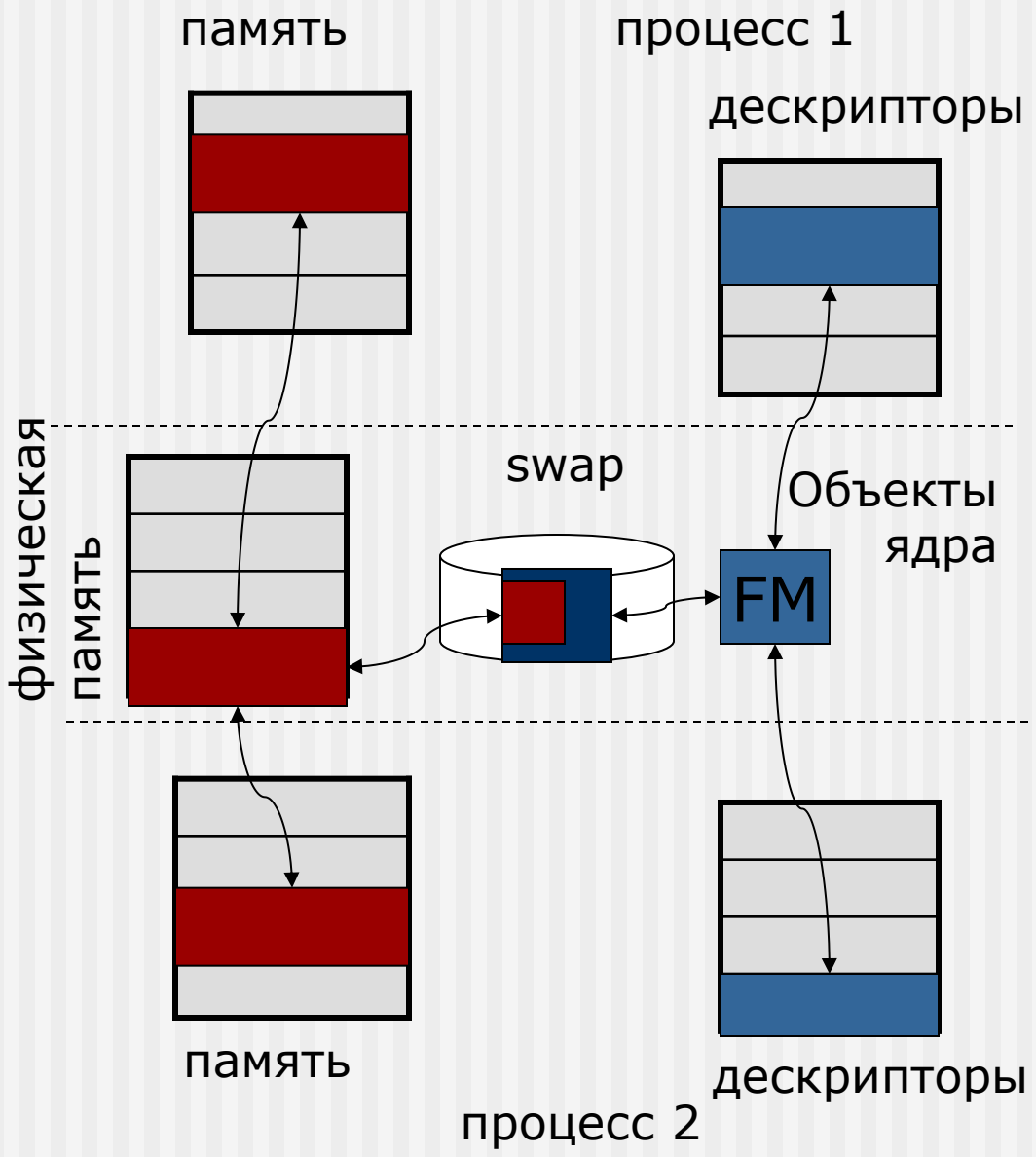
Создание разделяемой памяти

Процесс 1

- **Создать объект проекция файла (hFile=NULL, задать размер и имя)**
- **Спроецировать память**

Процесс 2

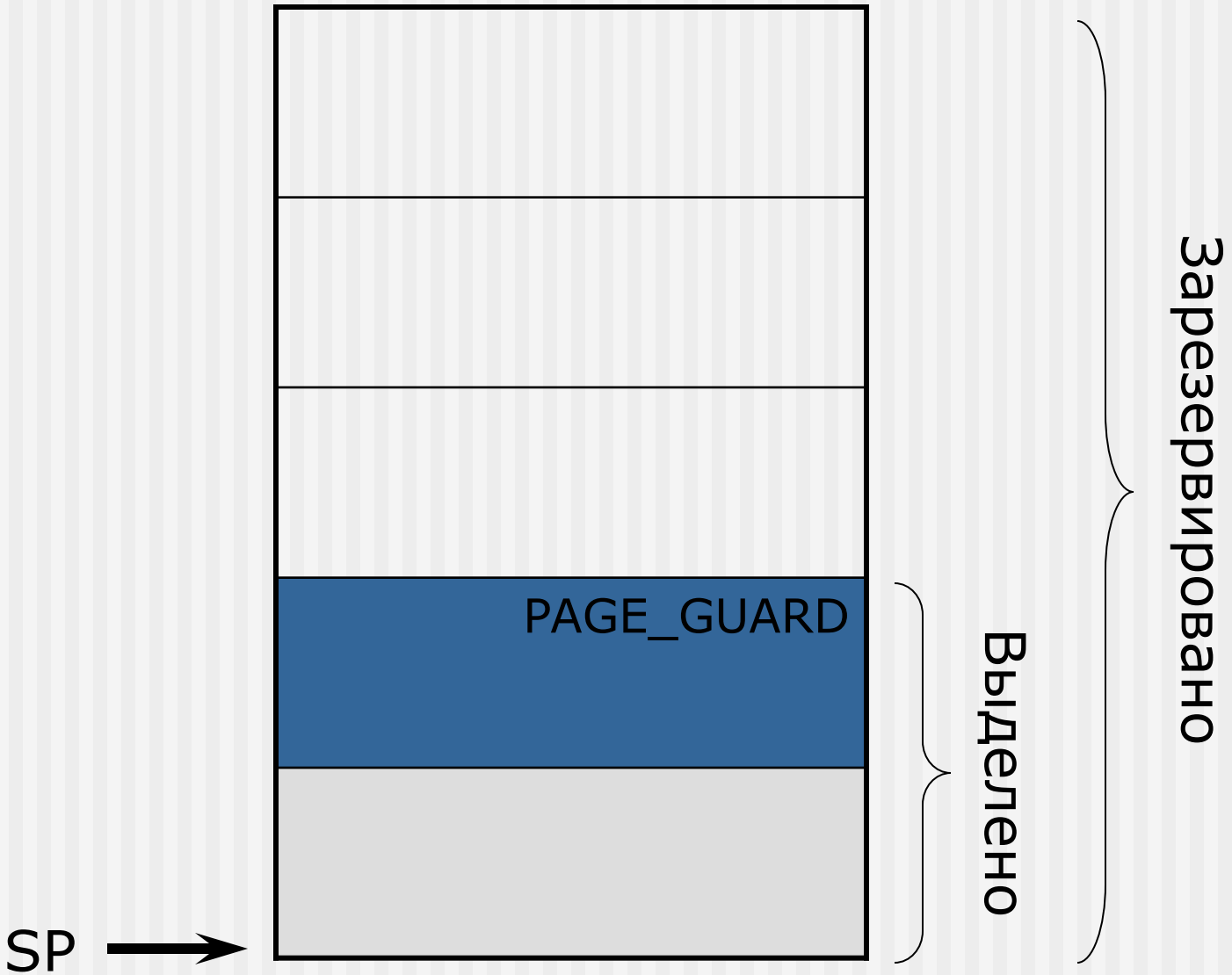
- **Открыть объект проекция файла**
- **Спроецировать память**



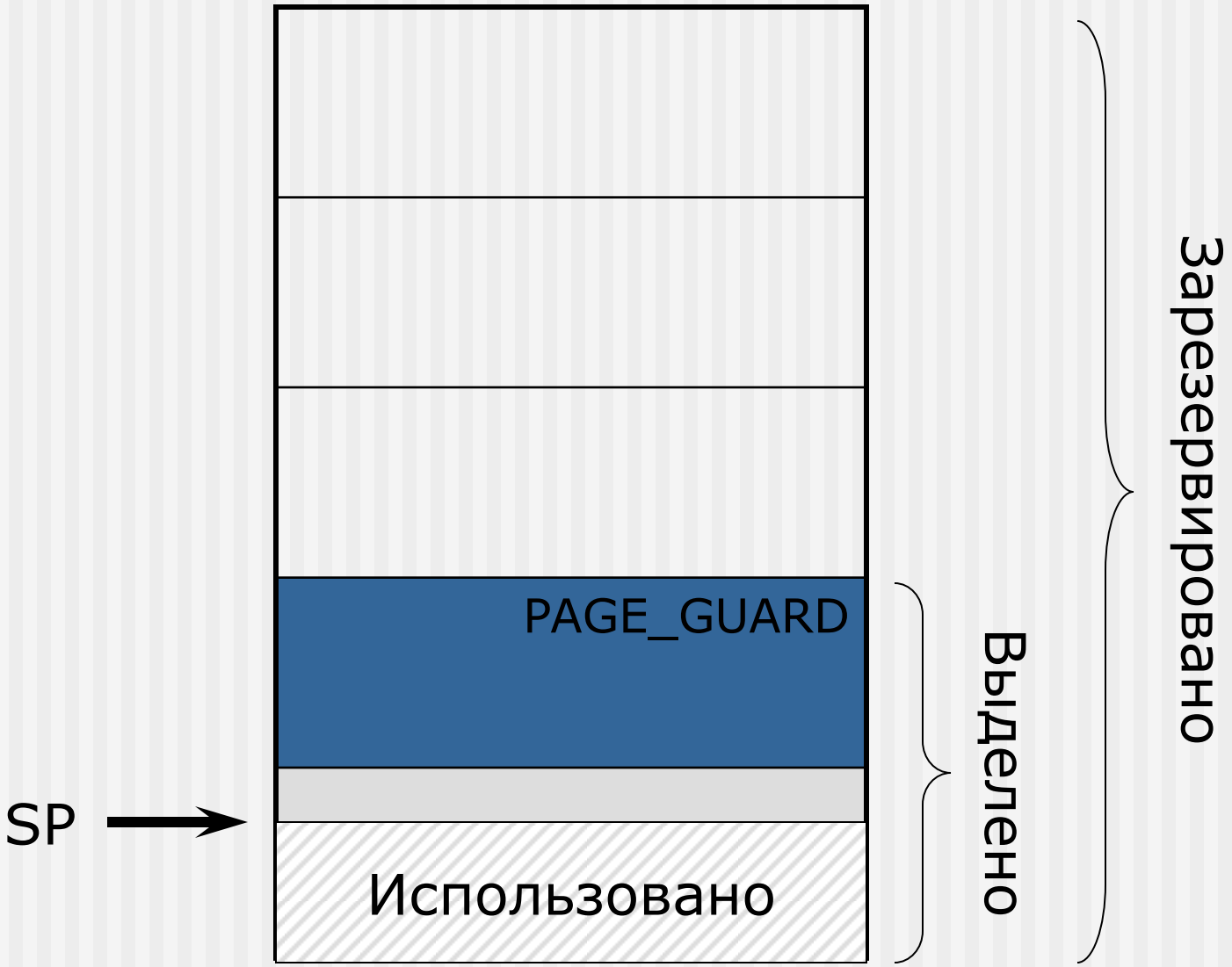
Стек процесса и флаг PAGE_GUARD

1. Резервируется регион максимального размера
2. Выделяются две страницы, на второй устанавливается флаг PAGE_GUARD
3. При обращении к странице с флагом PAGE_GUARD
    1. Происходит исключение
    2. Выделяется следующая страница
    3. Флаг PAGE_GUARD переставляется на последнюю выделенную страницу

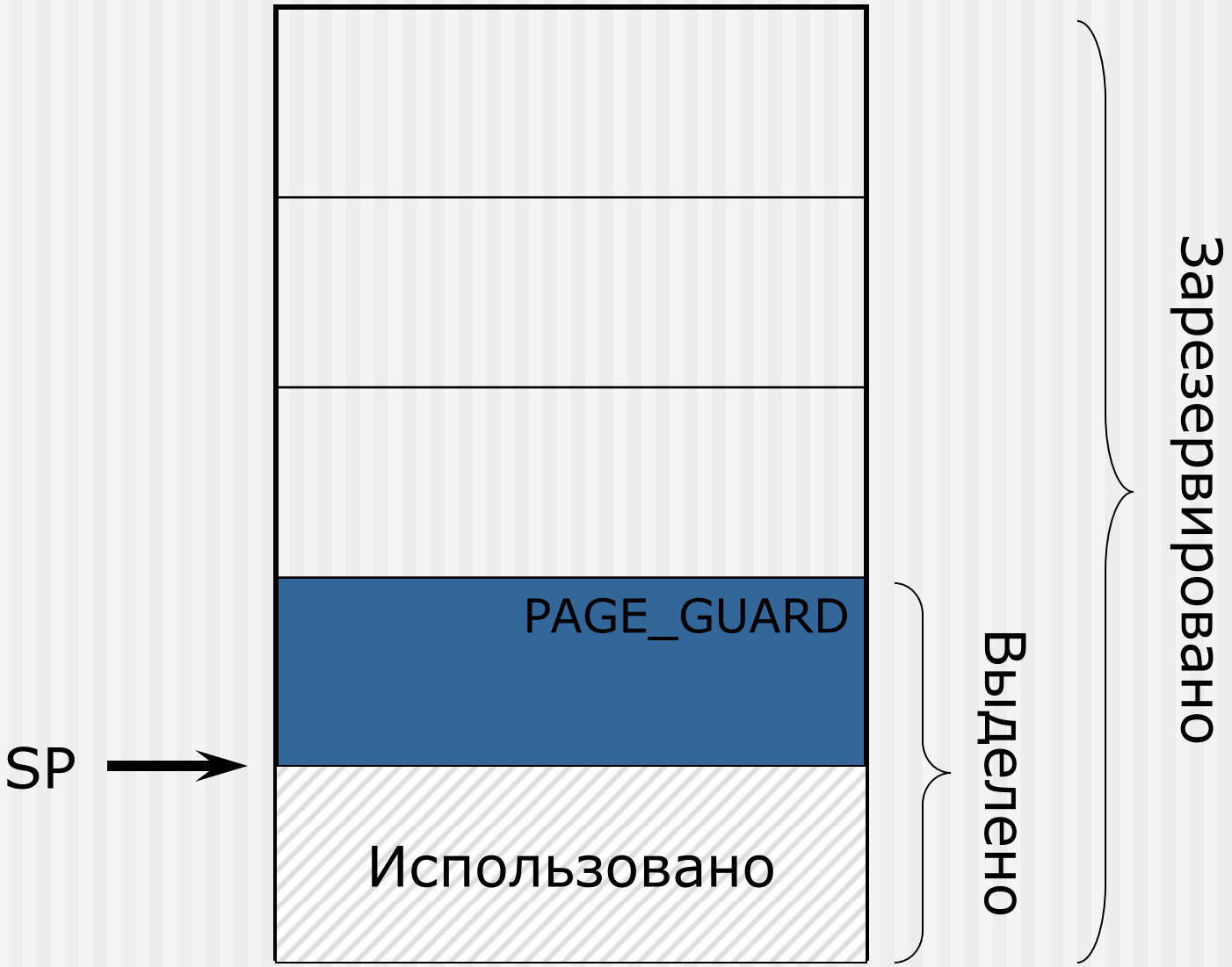
Стек процесса и флаг PAGE_GUARD



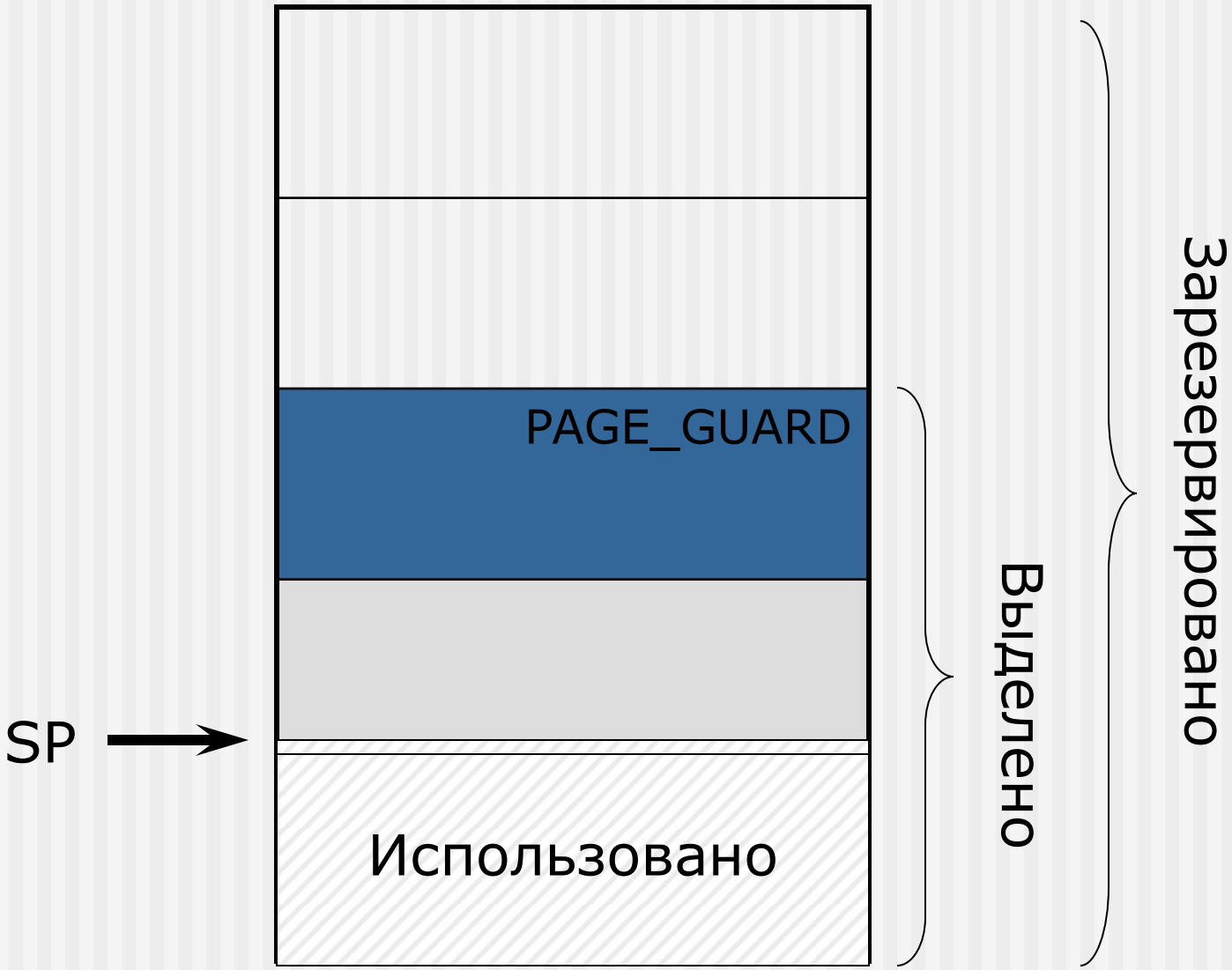
Стек процесса и флаг PAGE_GUARD



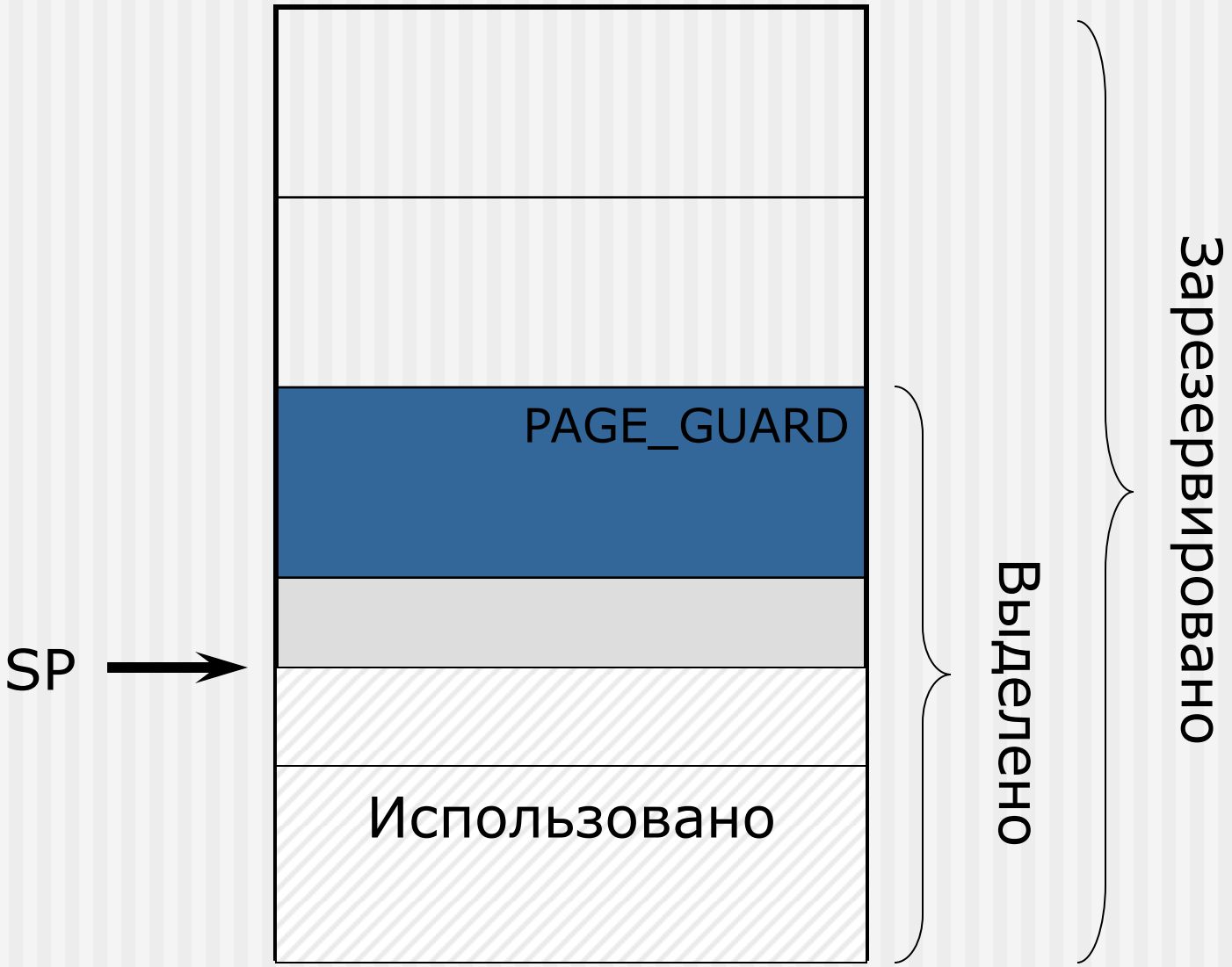
Стек процесса и флаг PAGE_GUARD



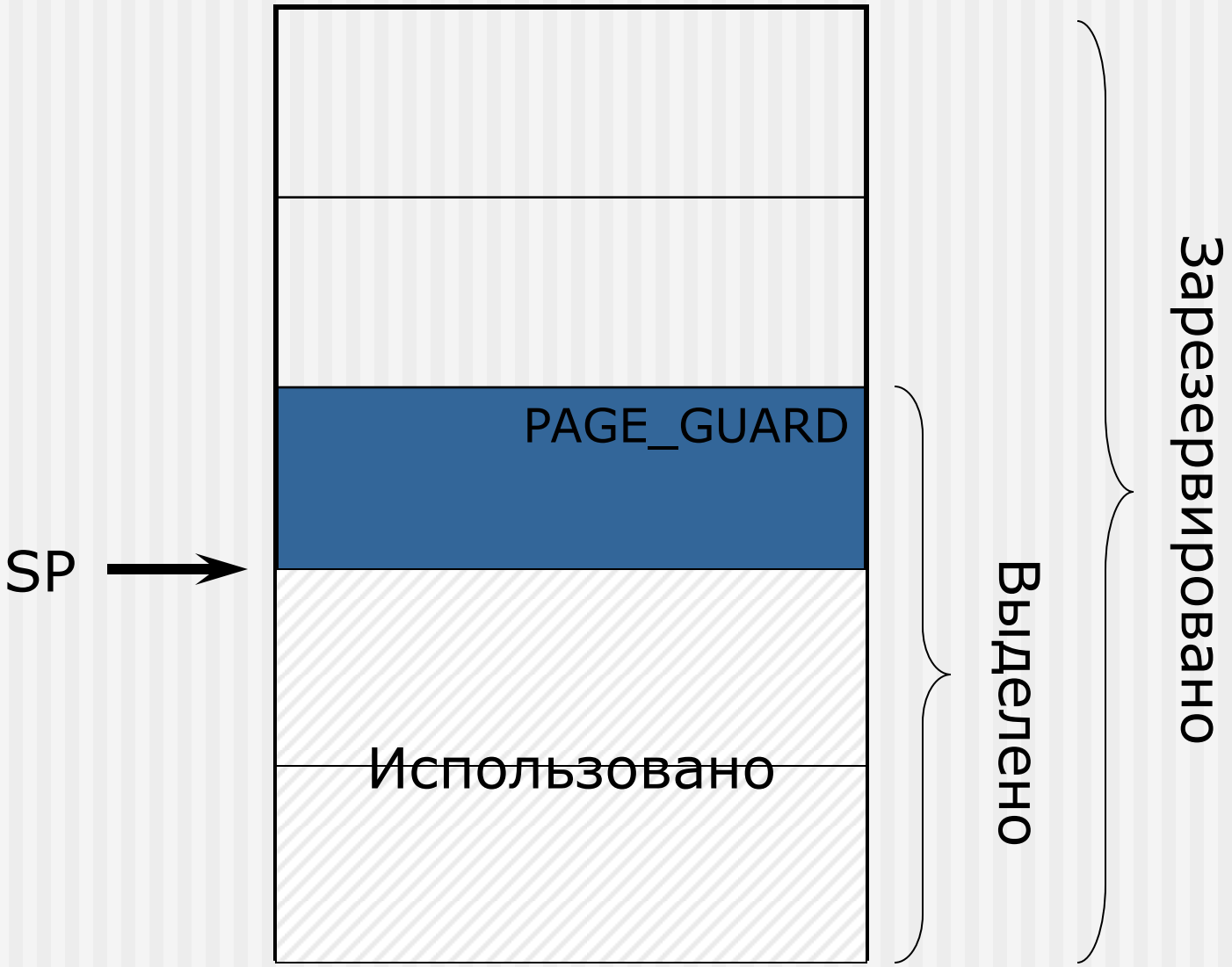
Стек процесса и флаг PAGE_GUARD



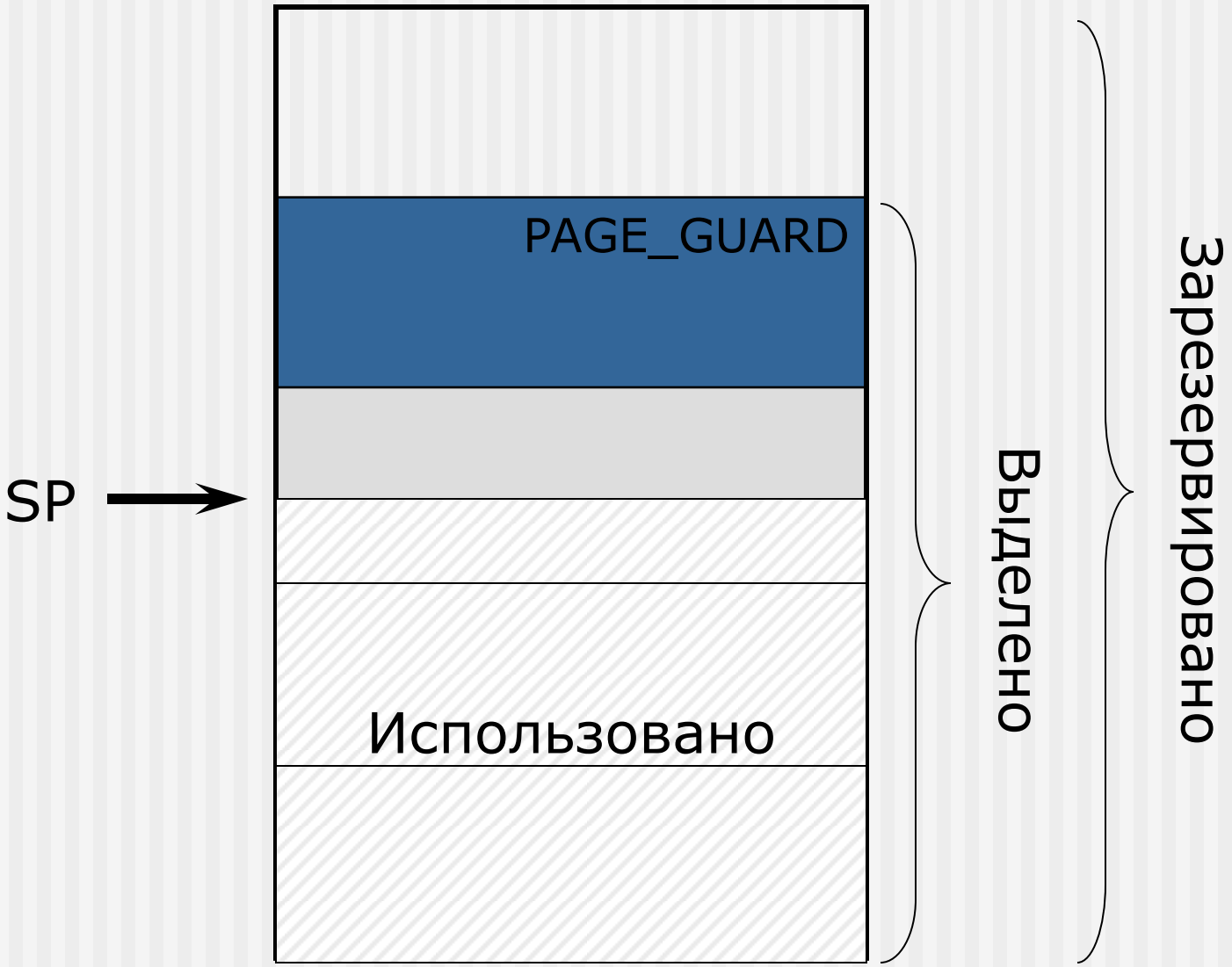
Стек процесса и флаг PAGE_GUARD



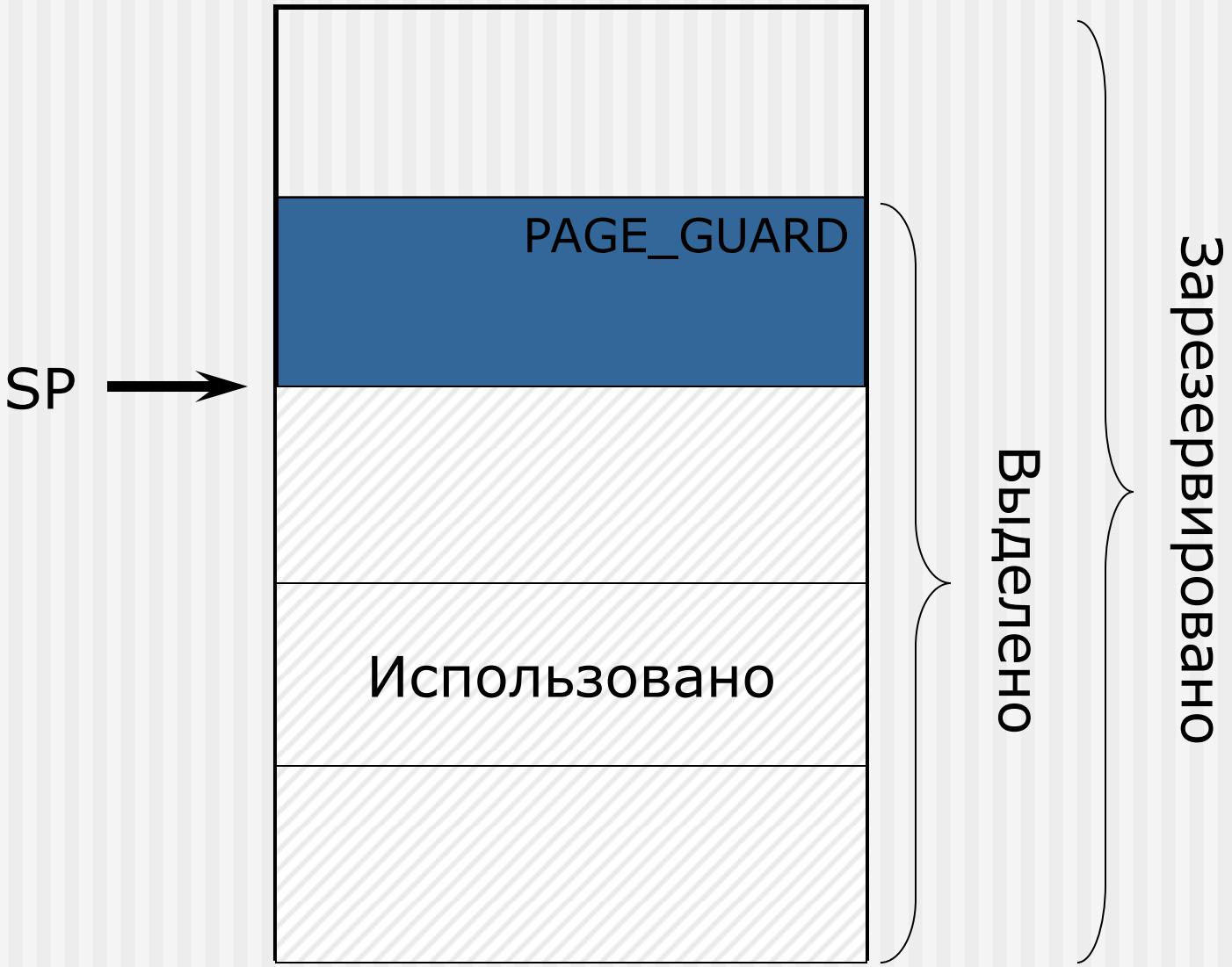
Стек процесса и флаг PAGE_GUARD



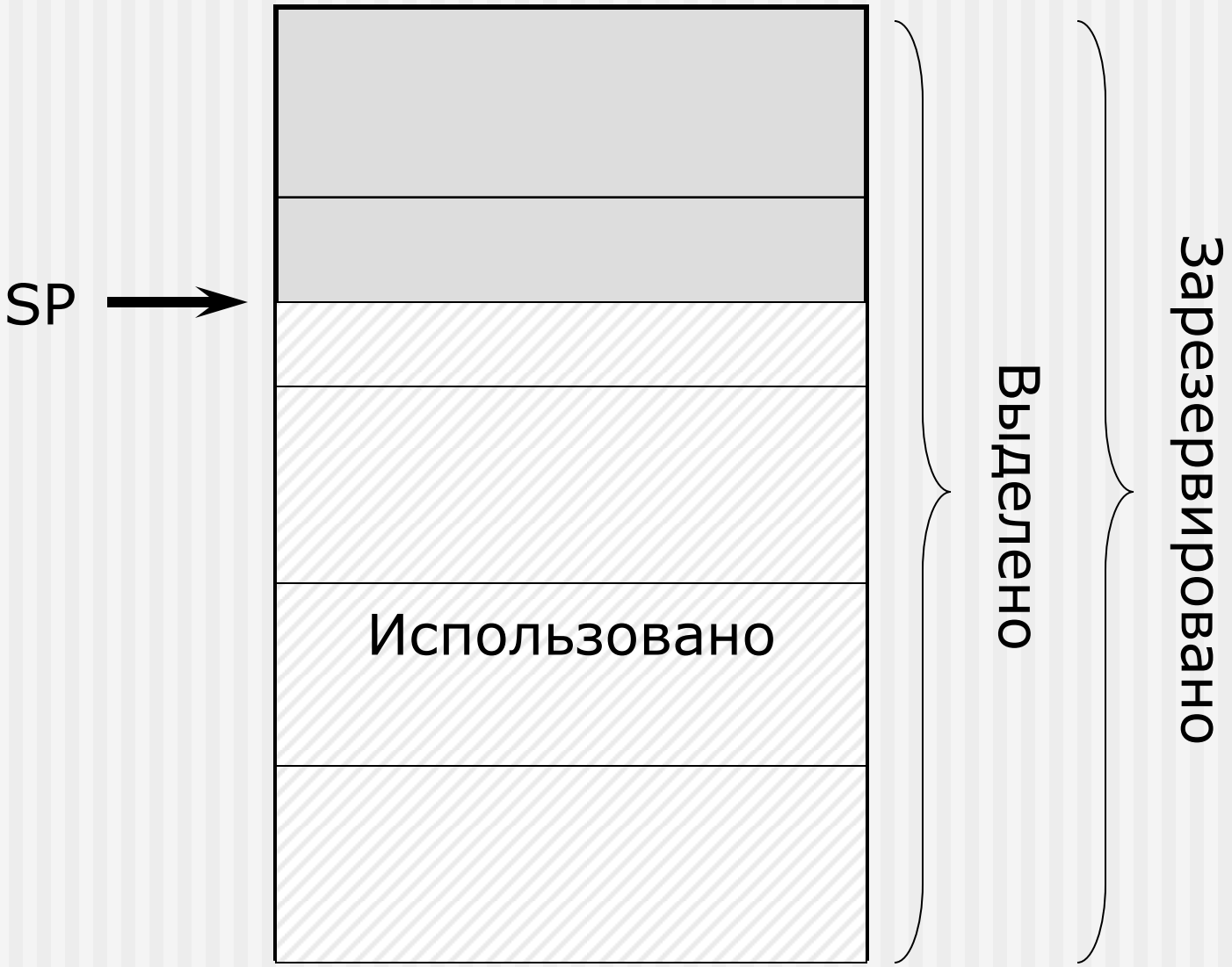
Стек процесса и флаг PAGE_GUARD



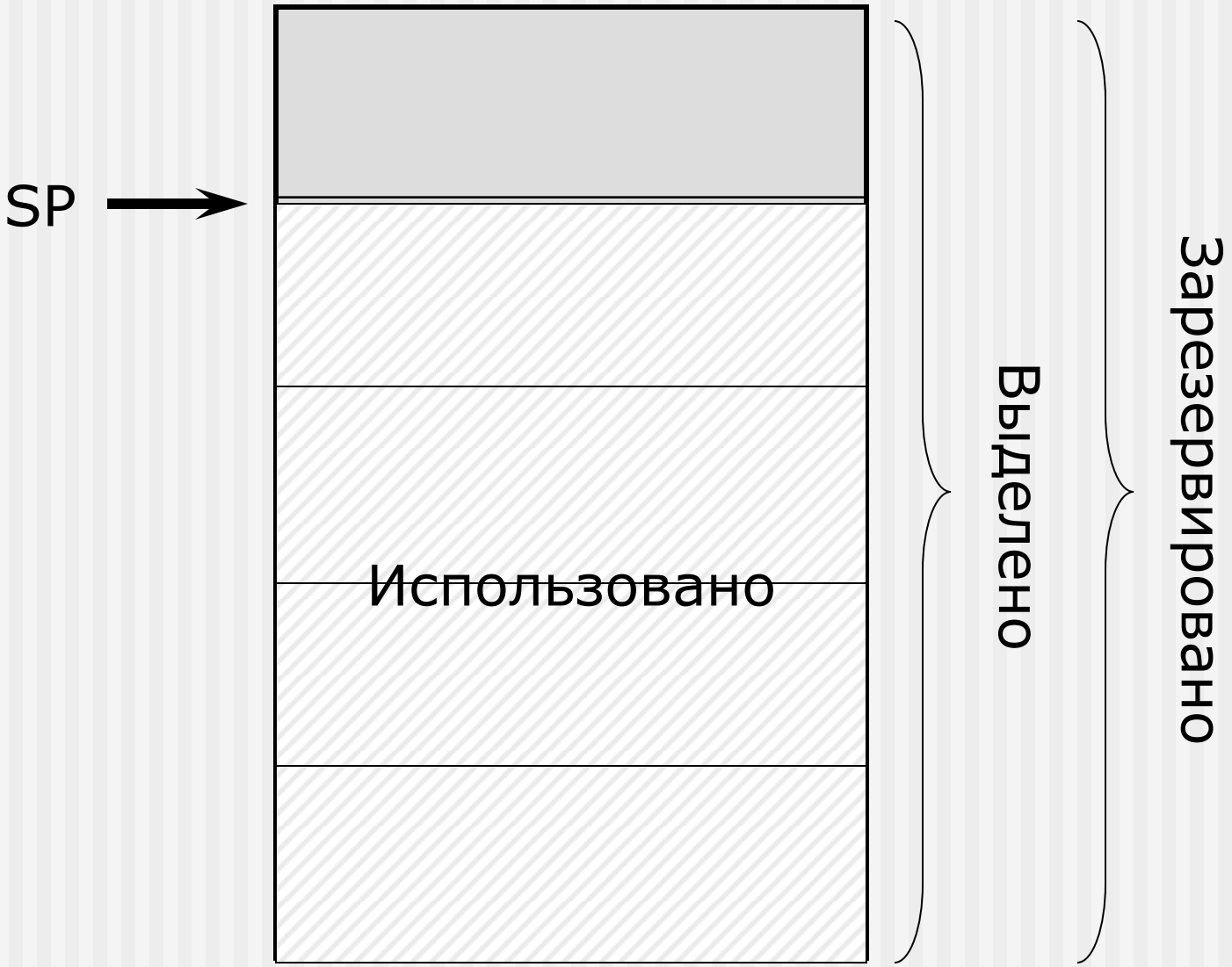
Стек процесса и флаг PAGE_GUARD



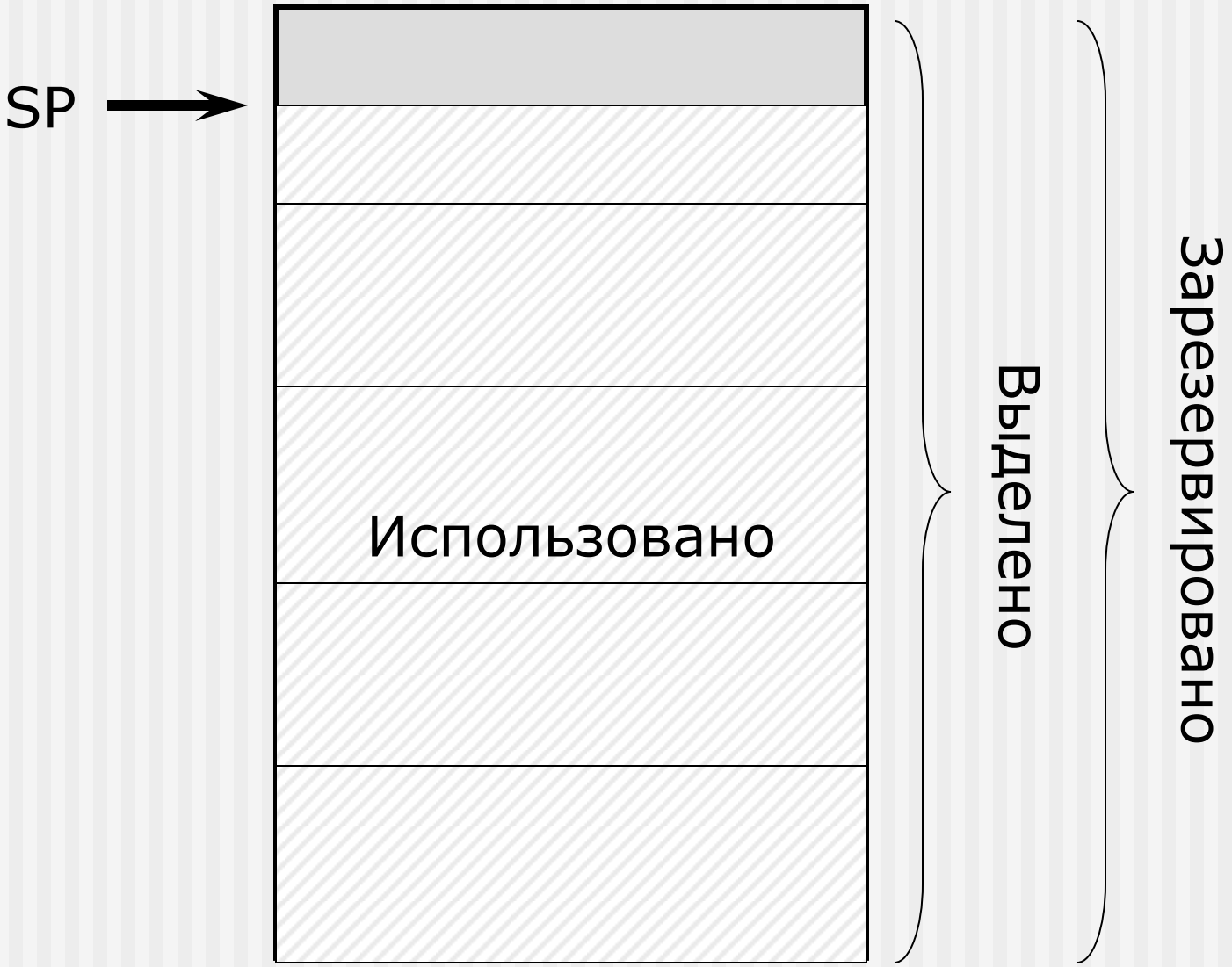
Стек процесса и флаг PAGE_GUARD



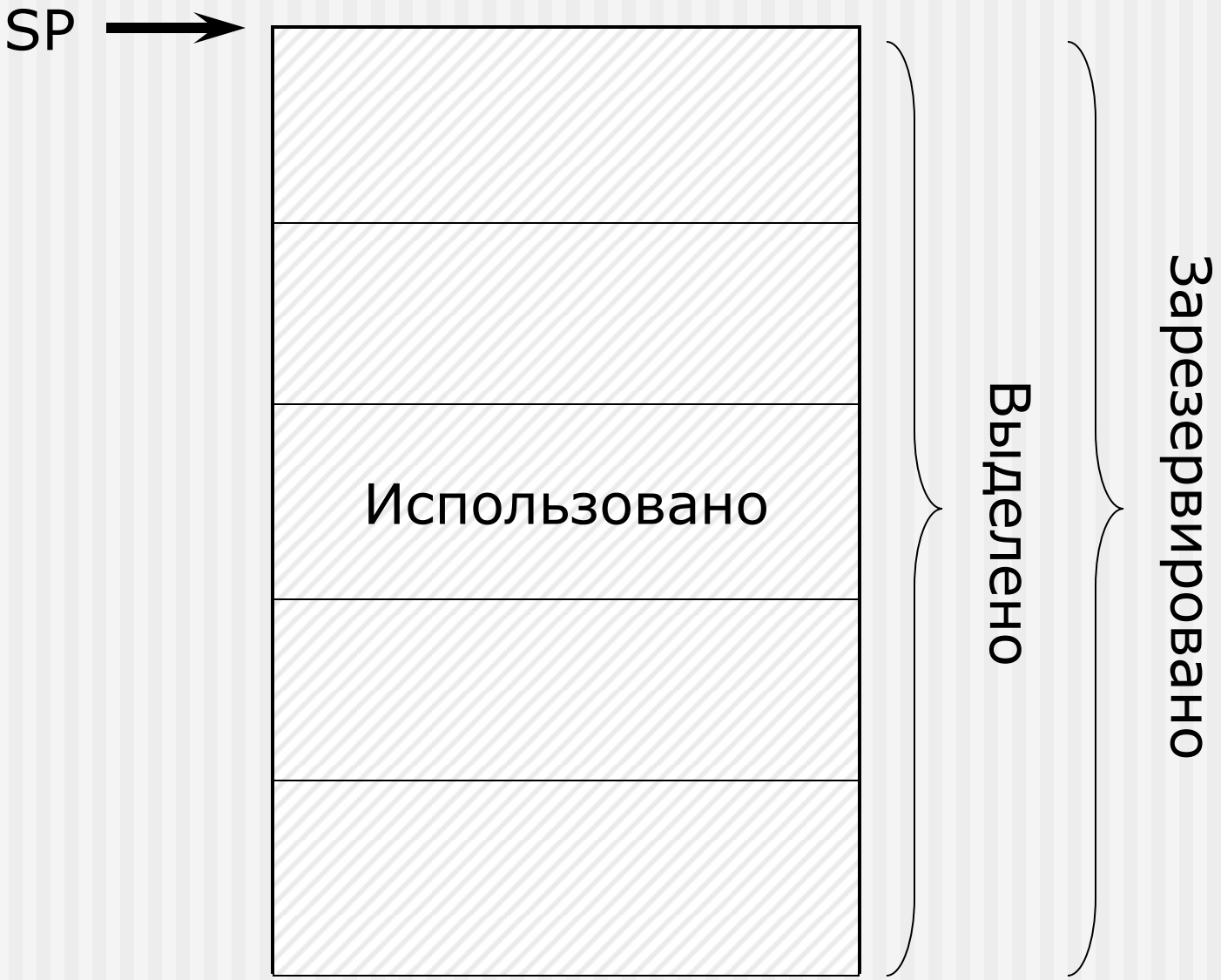
Стек процесса и флаг PAGE_GUARD



Стек процесса и флаг PAGE_GUARD



Стек процесса и флаг PAGE_GUARD

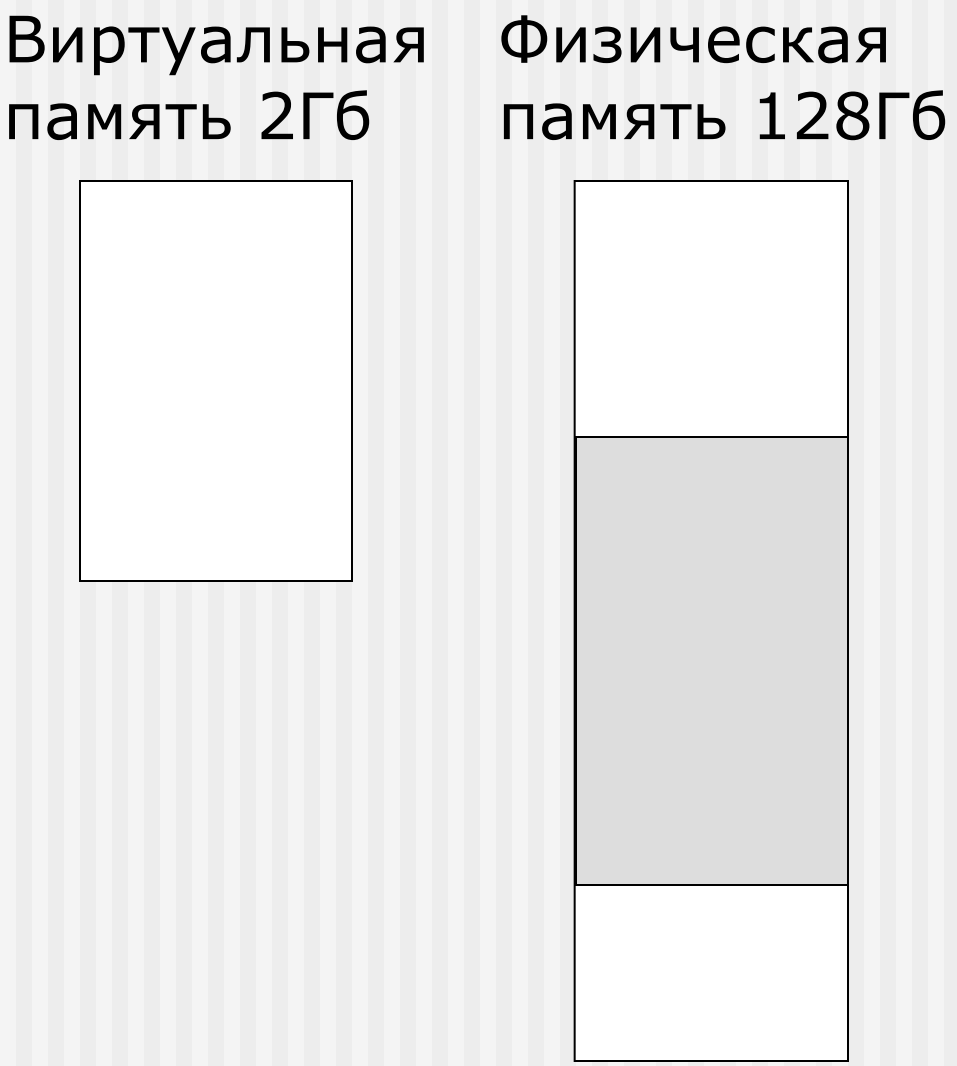


Address Windowing Extensions / AWE

- Выделение физической памяти
- Создание региона виртуальной памяти – окна
- Проецирование на окно физической памяти

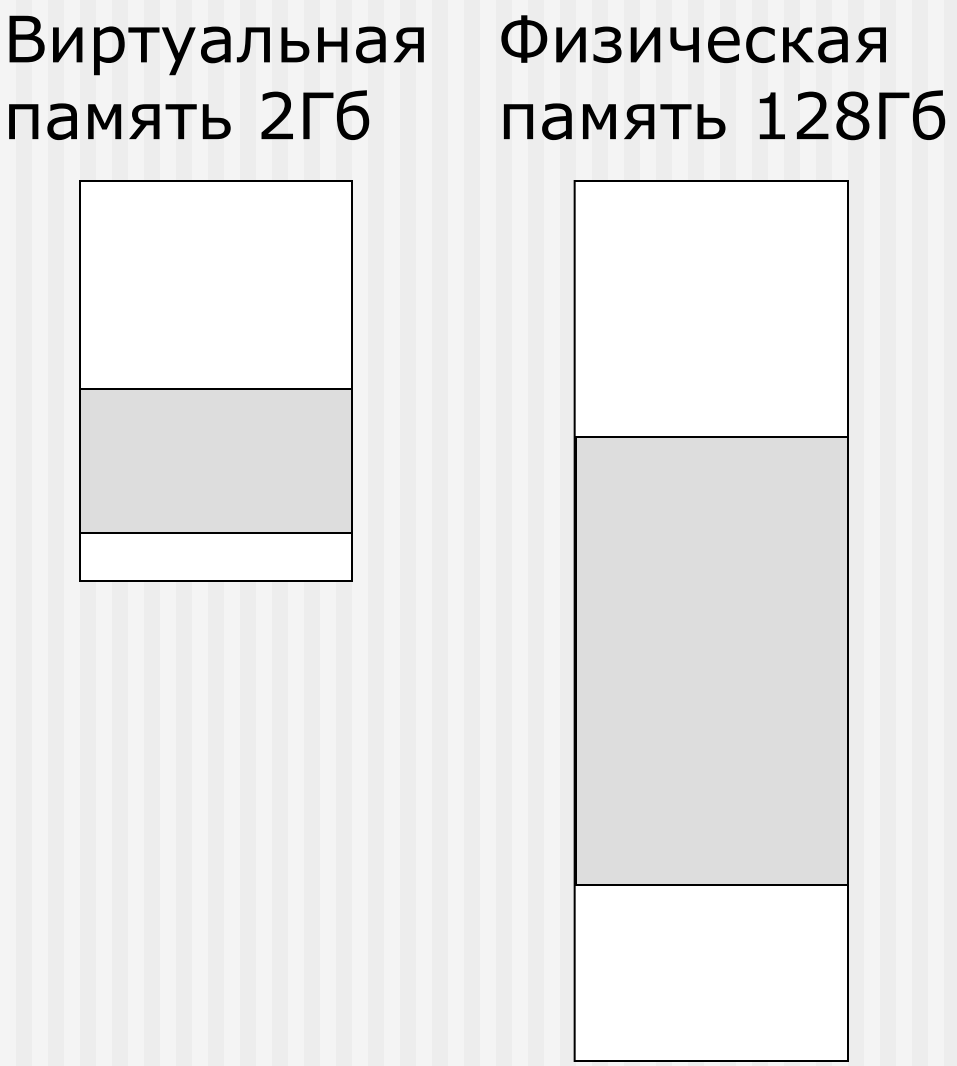
Address Windowing Extensions / AWE

- **Выделение физической памяти**
- Создание региона виртуальной памяти – окна
- Проецирование на окно физической памяти



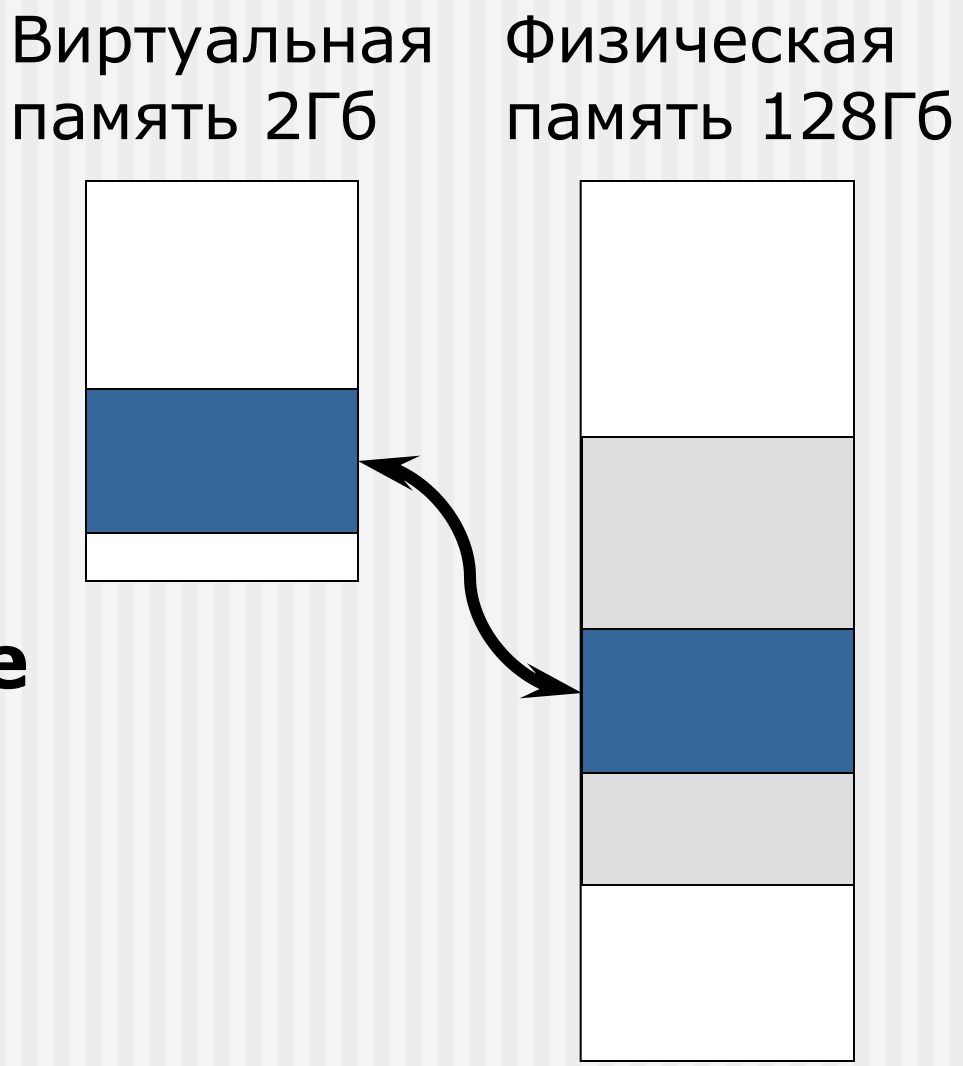
Address Windowing Extensions / AWE

- **Выделение физической памяти**
- **Создание региона виртуальной памяти – окна**
- Проецирование на окно физической памяти



Address Windowing Extensions / AWE

- **Выделение физической памяти**
- **Создание региона виртуальной памяти – окна**
- **Проецирование на окно физической памяти**



Address Windowing Extensions / AWE

- **Выделение физической памяти**
- **Создание региона виртуальной памяти – окна**
- **Проецирование на окно физической памяти**