1. Что такое программа?

**Программа** – это данные, предназначенные для управления конкретными компонентами системы обработки информации (СОИ) в целях реализации определенного алгоритма

1. Что такое программное обеспечение?

**Программное обеспечение** (**ПО**) – совокупность программ СОИ и программных документов, необходимых для их эксплуатации

1. Какое бывает ПО? Пояснить каждый вид.

* Системное ПО
* **Промежуточное (связующее) ПО**
* Прикладное ПО

**Промежуточное (связующее) ПО** – совокупность программ, осуществляющих управление программными ресурсами, порожденными программами и ориентированными на решение широкого спектра задач; комплекс технологического ПО для обеспечения взаимодействия между различными приложениями, системами, компонентам.

**Системное ПО** — это совокупность программ, которые обеспечивают управление компонентами компьютера.

Существуют следующие группы Системного ПО:

* Операционные системы
* Интерфейсные оболочки (ОС)
* Системы управления файлами
* Системы программирования
* Утилиты
* Драйверы
* Средства сетевого доступа

Классификация системного ПО:

**Управляющее ПО** – системные программы, реализующие набор функций, который включает в себя управление ресурсами и взаимодействие с внешней средой СОИ, восстановление работы системы после проявления неисправностей в технических средствах

**Обслуживающее ПО (утилиты)** – программы, предназначенные для оказания услуг общего характера пользователям и обслуживающему персоналу СОИ

**Базовое системное ПО** – минимальный набор программных средств, обеспечивающий работу компьютера и компьютерной сети

**Сервисное системное ПО** – программы и программные комплексы, которые расширяют возможности базового ПО и организуют удобную среду для работы других программ и пользователя

**Прикладное ПО** — класс программ, предназначенный для решения практических задач и предназначенный на непосредственное взаимодействие с пользователями.

1. Назовите функции системного ПО?

* Создание операционной среды функционирования для программ
* Автоматизация разработки новых программ
* Обеспечение надежной и эффективной работы компьютера и компьютерной сети
* Проведение диагностики и профилактики аппаратуры компьютера и компьютерных сетей
* Выполнение вспомогательных технологических процессов (копирование, архивирование, восстановление после сбоев и т.д.)

1. Две классификации системного ПО. Пояснить.

В 3 вопросе

1. Что такое системное программирование?

**Системное программирование** – это процесс разработки системного ПО; это процесс разработки программ сложной структуры

1. Что такое система программирования? Что она в себя включает?

**Система программирования** – набор специализированных программ, которые выступают инструментальными средствами разработчика для полной поддержки процессов совместной разработки, доступа к коду, проектирования, разработки, отладки и тестирования создаваемых программ, их развертывания.

Системы программирования включают в себя следующие средства:

* Редактор текста
* Транслятор
* Компоновщик
* Отладчик
* Библиотеки подпрограмм

1. Что такое исходный модуль?

**Исходный модуль** – программный модуль на исходном языке, обрабатываемый транслятором и представляемый для него как целое, достаточное для проведения трансляции

1. Что такое транслятор? Какие существуют виды трансляторов?

**Транслятор** – системная программа, преобразующая исходную программу на одном языке программирования в программу на другом языке

Виды трансляторов:

* Ассемблер
* Компилятор
* Интерпретатор
* Эмулятор
* Перекодировщик
* Макропроцессор

1. Назовите этапы подготовки программы.

Компилятор:

**Шаг первый – Предварительная обработка кода**:

* Присоединение исходных файлов
* Работа макропроцессоров

**Шаг второй – Анализ**:

* Лексический анализ
* Синтаксический анализ
* Семантический анализ

**Шаг третий – Синтез**:

* Генерация машинно-независимого кода
* Оптимизация машинно-независимого кода
* Распределение памяти
* Генерация машинного кода
* Оптимизация машинного кода

Результатом работы компилятора является объектный модуль.

**Объектный модуль** – программный модуль, получаемый в результате трансляции исходного модуля. Содержимое объектного модуля не содержит признаков на каком языке был написан исходный модуль.

Компоновщик:

Поскольку транслятор обрабатывает только один конкретный модуль, он не может должным образом обработать те части этого модуля, в которых запрограммированы обращения к данным или процедурам, определенным в другом модуле

**Компоновщик** (Редактор связей) соединяет вместе все объектные модули, входящие в программу

**Загрузочный модуль** – программный модуль, представленный в форме, пригодной для загрузки в оперативную память для выполнения

1. Что такое объектный и загрузочный модули?

10 вопрос

1. Что такое операционная система?

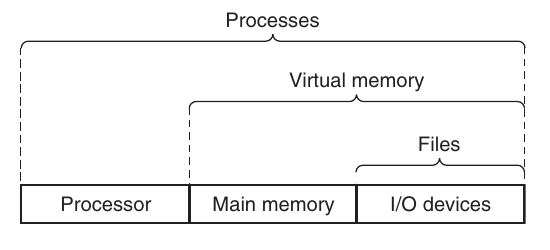
**Операционная система** – комплекс системного программного обеспечения, который предоставляет полезные абстракции базовых аппаратных средств

1. Основные предназначения ОС?

Операционная система имеет два основных предназначения:

* Защита аппаратного обеспечения от неправильного использования неконтролируемыми приложениями
* Предоставление приложениям простого и единообразного механизма для управления сложными и зачастую широко разнообразными низкоуровневыми аппаратными устройствами

1. Какие абстракции и над каким аппаратным обеспечением вводит ОС для разработчика?



1. Что такое ядро ОС?

**Ядро ОС** состоит из набора функций, предоставляющих фундаментальные механизмы системы. К их числу относятся сервисные функции планирования потоков и синхронизации, используемые исполнительными компонентами, и низкоуровневая поддержка, зависящая от аппаратной архитектуры, - диспетчеризация прерываний и исключений, зависящая от архитектуры процессора

Ядро операционной системы — это самая низкоуровневая часть программного обеспечения, которая представляет собой основу ОС и является связующим звеном между аппаратным и программным обеспечением компьютера.

1. Каковы задачи ядра ОС?

* Диспетчеризация процессов
* Управление памятью
* Предоставление файловой системы
* Создание и завершение процессов
* Доступ к устройствам
* Работа в сети
* Предоставление интерфейса прикладного программирования (API) системных вызовов

1. Что такое пользовательский режим и режим ядра?

Чтобы пользовательские приложения не могли прочитать критические данные операционной системы и/или изменить их, в Windows/Linux предусмотрены два режима доступа к процессору: **пользовательский режим** (user mode) и **режим ядра** (kernel mode).

1. Что такое системный вызов?

**Системный вызов (system call)** – представляет собой управляемую точку входа в ядро, позволяющую процессу запрашивать у ядра осуществления некоторых действий в интересах процесса**;** вызов функции ядра ОС прикладной программой

1. Что такое ловушки?

**Ловушки (traps)** – представляет собой неуправляемую точку входа в ядро, например, запросы, вызванные ошибкой деления на ноль и т.п. Обработка ловушек происходит в рамках программы вызвавшей такое поведение

1. Что такое прерывания?

**Прерывания (interrupts)** – представляют собой запросы к ядру ОС от внешних аппаратных устройств. Обрабатываются независимо от каких-либо программ пользователя

1. Что такое объекты ядра ОС?

Каждый **объект ядра** – на самом деле просто блок памяти, выделенный ядром и доступный только ему

1. Какие объекты вы знаете?

В частности, в Windows объектами ядра являются: маркеры доступа (access token objects), файлы (file objects), проекции файлов (file-mapping objects), порты завершения ввода-вывода (I/O completion port objects), задания (job objects), почтовые ящики (mailslot objects), мьютексы (mutex objects), каналы (pipe objects), процессы (process objects), семафоры (semaphore objects), потоки (thread objects) и ожидаемые таймеры (waitable timer objects)

1. Что такое дескриптор?

**Дескриптор** — это специальный идентификатор, который операционная система использует для управления доступом к различным объектам ядра, таким как процессы, потоки, файлы и события.

1. Что такое POSIX?

**POSIX** – набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов

1. Что такое процесс?

**Процесс** – управляющий объект, который обеспечивает изоляцию адресных пространств и представляет работающий экземпляр программы; исполняемое на устройстве приложение вместе со всеми ресурсами, которые требуются для его исполнения

1. Какие ресурсы процесса вам известны?

* Образ исполняемого файла
* Память
* Список дескрипторов объектов, выделенных процессу (файлы, потоки, объекты синхронизации и т.д.)
* Атрибуты безопасности (маркеры)
* Контекст процесса

1. Что такое контекст процесса? Для чего он нужен?

**Контекст процесса** – минимальный набор данных, используемый процессом, который должен быть сохранен, чтобы выполнение процесса могло быть прервано и в последующем возобновлено с той же точки

Контекст процесса состоит из:

* Program Counter (PC) или Instruction Pointer (IP)
* Значений регистров
* Стека
* Кучи
* Глобальных переменных
* и т. д.

1. Назовите системные вызовы для создания процесса в Windows. В Linux (POSIX)

Системный вызов Windows – **NtCreateUserProcess**

Системные вызовы Linux (POSIX) – **fork**, **vfork**

1. Назовите пользовательские функции, с помощью которых можно создать процесс в Windows. В Linux (POSIX)

Windows: Функции[**CreateProcessA**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa)и[**CreateProcessW**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessw)

Linux: Функции [**fork**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/fork.html) и [**exec**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/exec.html)

Функция **system** ([Win](https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/system-wsystem?view=msvc-170)/[POSIX](https://pubs.opengroup.org/onlinepubs/9799919799/functions/system.html)) – позволяет вызывающей программе выполнять произвольные консольные команды

Функция **fork** – предназначена для создания нового дочернего процесса, который будет являться полной (насколько это возможно) копией родительского процесса

Функция **exec** – предназначена для замены образа исполняемого файла в рамках существующего процесса

Важно! Никаких новых процессов при вызове **exec** не создаётся. Загружается новая программа!

1. Назовите функции и их назначения, которые предназначены для работы с процессами в Windows. В Linux (POSIX)

Перед завершением процесса может возникнуть ситуация, когда требуется дождаться завершения другого (напр. дочернего) процесса. В таком случае могут помочь функции [**WaitForSingleObject**](https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitforsingleobject) и [**WaitForMultipleObjects**](https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitformultipleobjects).

**ExitProcess:** Является функцией для **штатного** завершения работы процесса. Происходит выгрузка всех ресурсов, используемых процессом

**TerminateProcess:** Является функцией для **аварийного** завершения работы процесса. Освобождение ресурсов процесса **не гарантируется**. Рекомендуется использовать для завершения **зависших** процессов

**CloseHandle:**

* Функция делает указанный дескриптор не действительным
* Уменьшает значение счётчика дескрипторов указанного объекта
* Производит проверку на то, требуется ли освобождение объекта из системы
* Использование данной функции на дескрипторе процесса не приводит к завершению его работы
* В случае вызова функции **TerminateProcess**,требуется вручную вызвать **CloseHandle** для дескриптора завершённого процесса
* **ExitProcess** вызывает данную функцию самостоятельно

Linux:

Функция [**wait**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/wait.html)– приостанавливает выполнение родителя, пока не будет завершен один из его потомков

Библиотечная функция [**exit**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/exit.html)завершает процесс, делая все его ресурсы (память, дескрипторы открытых файлов и т. д.) доступными для последующего перераспределения ядром.

1. Что такое межпроцессорное взаимодействие(IPC)?

**IPC** (Inter-process communication, рус., **межпроцессное взаимодействие**) – механизм, позволяющий процессам обмениваться данными и синхронизировать свои действия

Это фундаментальная концепция системного программирования, поскольку она позволяет процессам работать вместе для достижения общей цели

1. Какие категории IPC вам известны?

* Взаимодействие
* Синхронизация
* Сигналы

1. Какие механизмы входят в каждую категорию?

1. **Взаимодействие**

* Каналы (Pipes): Однонаправленный способ передачи данных между процессами. Данные записываются в один конец канала и читаются из другого.
* Сетевые сокеты: Позволяют процессам обмениваться данными через сеть, используя протоколы, такие как TCP и UDP.
* Сообщения (Message Queues): Позволяют процессам отправлять и получать сообщения в очереди, обеспечивая асинхронный обмен данными.
* Разделяемая память (Shared Memory): Область памяти, доступная нескольким процессам, позволяющая им обмениваться данными быстрее, чем через другие механизмы IPC.
* Файлы: Процессы могут обмениваться данными, записывая и читая из файлов на диске.

2. **Синхронизация**

* Мютексы (Mutexes): Позволяют только одному потоку или процессу иметь доступ к ресурсу в любой момент времени, предотвращая гонки данных.
* Семафоры: Счётчики, которые управляют доступом к ресурсу, позволяя нескольким процессам одновременно использовать ограниченное количество ресурсов.
* Мониторы: Обеспечивают защиту данных и синхронизацию доступа к ресурсам, комбинируя мютексы и условия.
* Условные переменные: Используются для сигнализации между потоками о том, что ресурс стал доступным или изменился.

3. **Сигналы**

* POSIX-сигналы: Сигналы, такие как SIGINT, SIGTERM, и SIGKILL, используются для управления процессами, например, для их завершения или назначения.
* Обработчики сигналов: Функции, которые могут быть зарегистрированы для обработки определённых сигналов, позволяя процессам реагировать на события.
* Сигналы пользовательского уровня: Позволяют процессам отправлять и обрабатывать собственные сигналы для коммуникации.

1. Опишите общие концепции передачи данных?

Процесс, который посылает данные другому потоку, называется **отправителем**

Процесс, который получает данные от другого потока, называется **адресатом** или **получателем**

С точки зрения направления передачи данных различают следующие виды связей:

* **Симплексная** – передача только, в одном направлении
* **Полудуплексная** – передача в обе стороны, но одновременно только в одну сторону
* **Дуплексная** – передача в двух направлениях одновременно

Под **топологией** связи будем понимать конфигурацию связей между процессами-отправителями и адресатами. С точки зрения топологии различают следующие виды связей: 1-1, 1-n, n-1, n-m.

Концептуально обмен данными между процессами выполняется при помощи двух функций: ***send*** – послать данные, ***receive*** – получить данные

При передаче данных может использоваться **прямая** или **косвенная** адресация процессов

При прямой адресации процессов в функциях send и receive явно указываются процессы отправитель и адресат

При косвенной адресации в функциях send и receive указываются не адреса, а имя связи, по которой передаются данные

Адресация процессов может быть **симметричной** и **асимметричной**. Если при обмене данными между процессами используется только прямая или только косвенная адресация, то такая адресация процессов называется симметричной. Если же при обмене данными между процессами используется как прямая, так и косвенная адресация, то такая адресация процессов называется асимметричной

При передаче данных различают **синхронный** и **асинхронный** обмен данными

Синхронная отправка – отправитель при отправке данных, блокируется до получения этих данных адресатом

Синхронное получение – адресат вызывая ***receive*** блокируется до тех пор, пока не получит данные

Асинхронные варианты данных операций не приводят к блокировке отправителя и адресата соответственно

(А)синхронный обмен = (А)синхронная отправка + (А)синхронное получение

В целом в системах может использоваться **смешанный** обмен данными

1. Сравните механизмы анонимных и именованных каналов

Анонимные каналы применяются только в рамках родительски-дочерних отношений между процессами, откуда следует, что данные каналы могут применяться только в рамках одного устройства

Именованные каналы могут применяться между независимыми процессами, а также могут использоваться между процессами на разных устройствах (по сети)

Именованные каналы обладают дополнительными свойствами:

* Могут быть дуплексными
* Передача данных может быть и поточная, и сообщениями
* Есть возможность асинхронного обмена данными
* Имеют имя формата: «\\.\pipe\pipe\_name**» (**[**Windows**](https://learn.microsoft.com/en-us/windows/win32/ipc/pipes)**)**, в Linux имя не имеет строгого формата

1. В какой ситуации предпочтительнее использовать каналы и очереди? Разделяемую память?

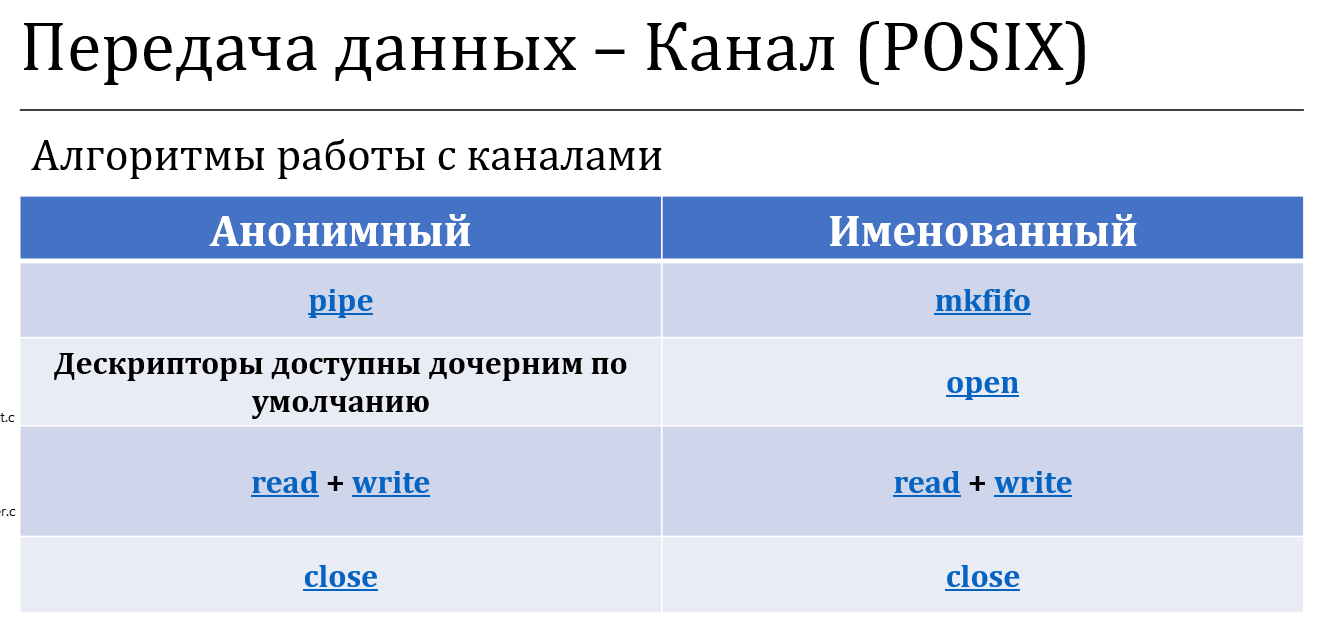
Для обработки данных в конвейерной структуре, когда один процесс передает вывод другому процессу, предпочтительнее выбрать каналы.

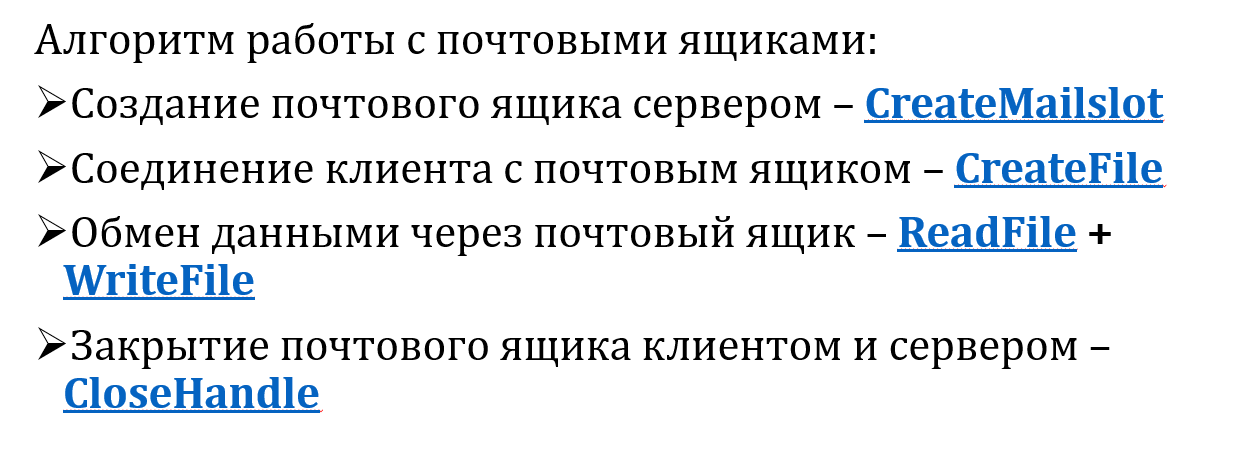
Если требуется обработка последующих запросов или важен приоритет сообщений, то очереди.

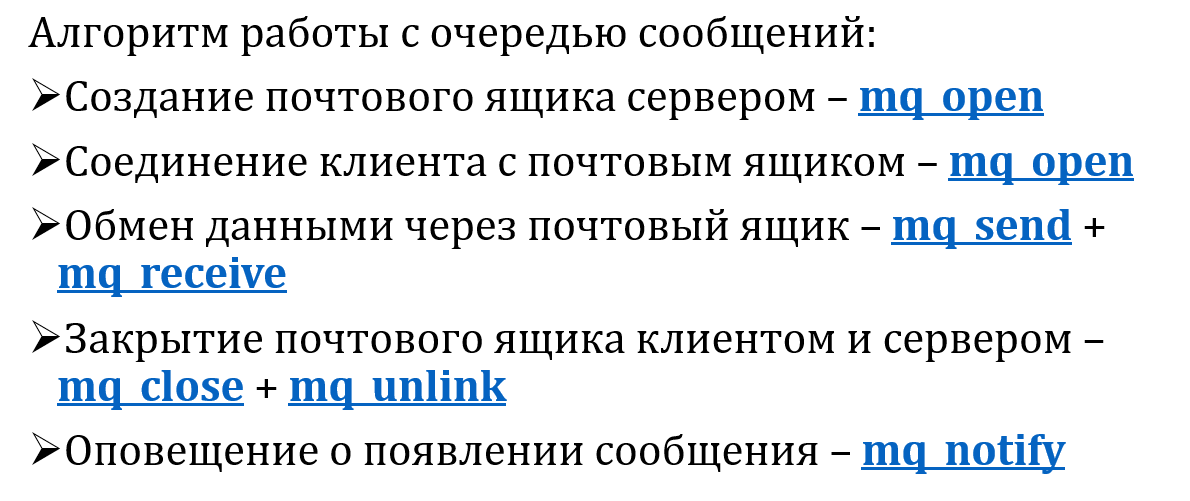
Разделяемую память предпочтительнее использовать в случае частого обмена большими объёмами данных

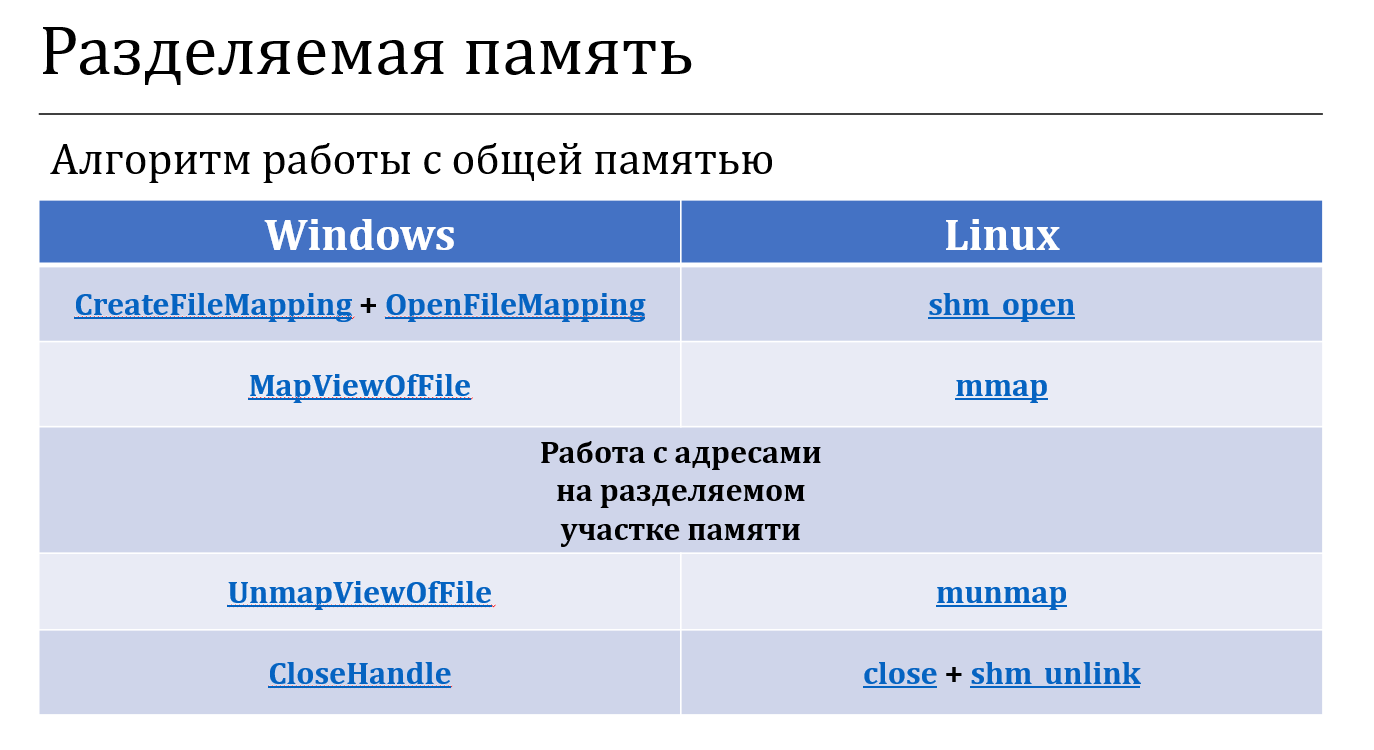
1. Назовите функции и их назначения, которые предназначены для работы с любым механизмом IPC в Windows. В Linux (POSIX).



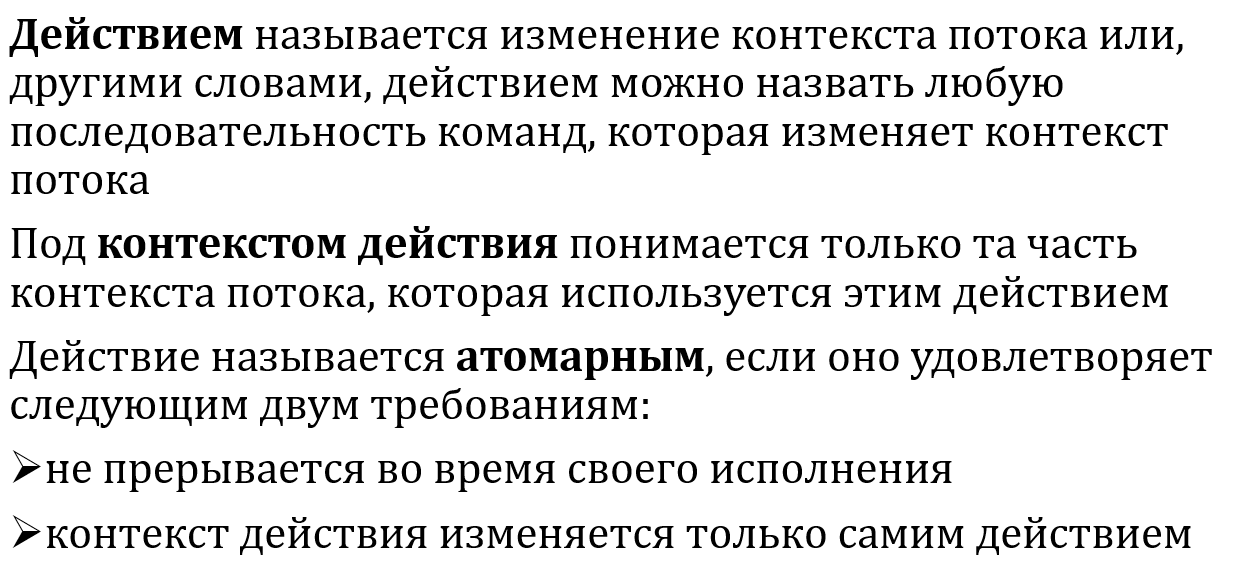




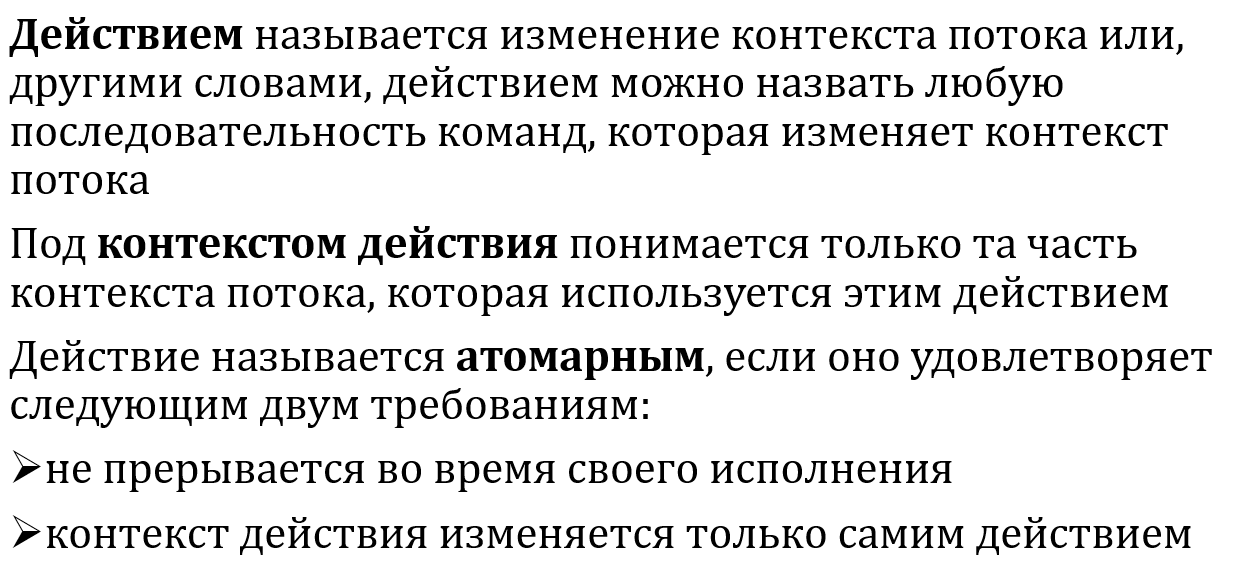


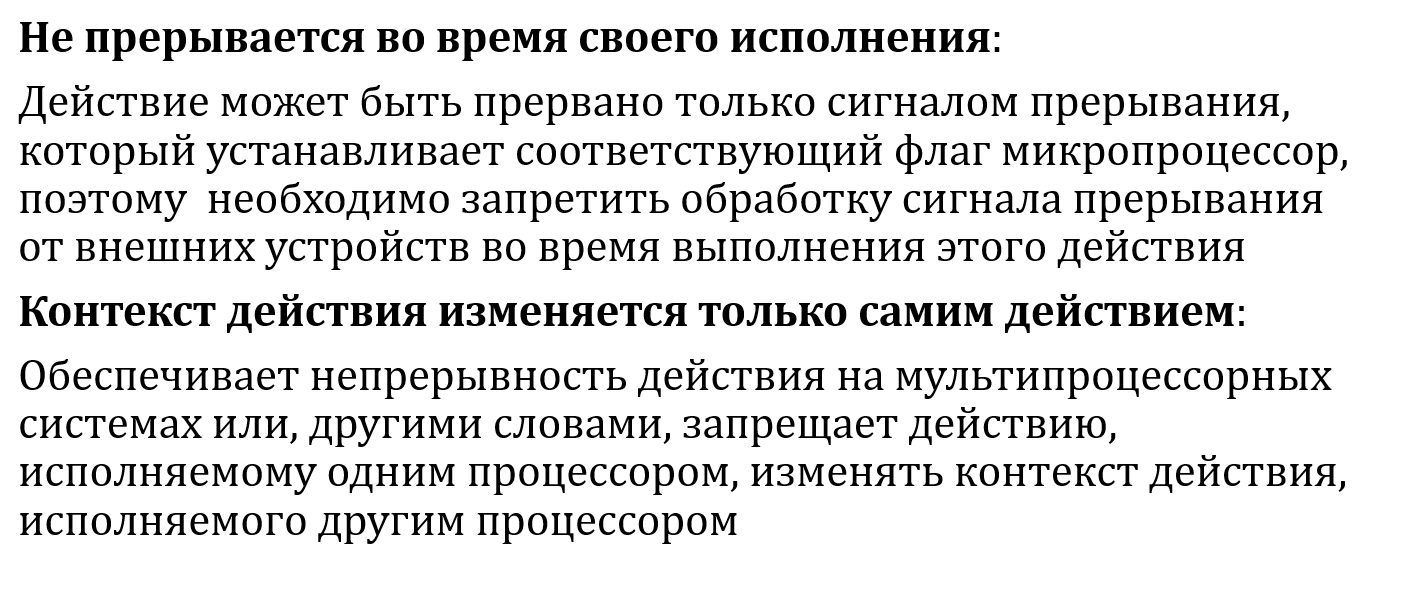


1. Что такое действие? Контекст действия?

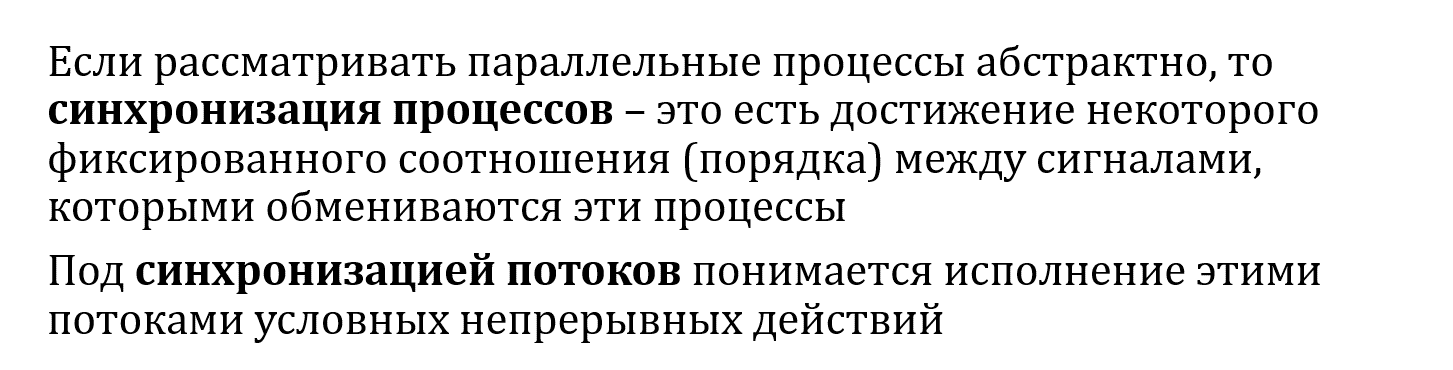


1. Что такое атомарное действие?





1. Что такое синхронизация процессов? Потоков?

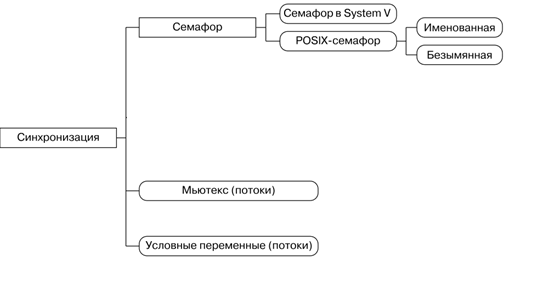


1. Какие механизмы синхронизации вам известны?

* Критическая секция
* Семафор
* Мьютекс
* Условные переменные
* События ([WinAPI](https://learn.microsoft.com/en-us/windows/win32/sync/synchronization-functions), WinAPI [UseCase](https://learn.microsoft.com/en-us/windows/win32/sync/using-event-objects), [POSIX API](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_cond_init.html))
* Барьеры (WinAPI [about](https://learn.microsoft.com/en-us/windows/win32/sync/synchronization-barriers), [POSIX API](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_barrier_destroy.html))
* Блокировка файлов
* Спинлок ()
* RW-лок ()
* Условные переменные ()
* Атомарные операции ()

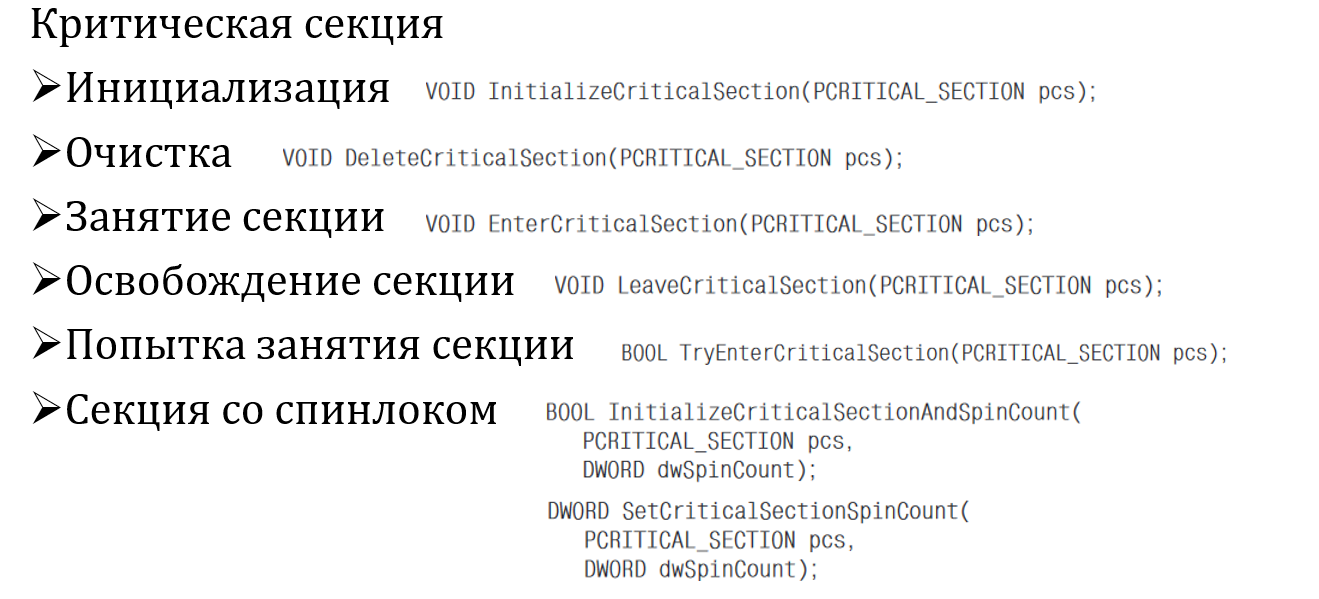
1. Какие механизмы лучше использовать для синхронизации потоков одного процесса? Почему?

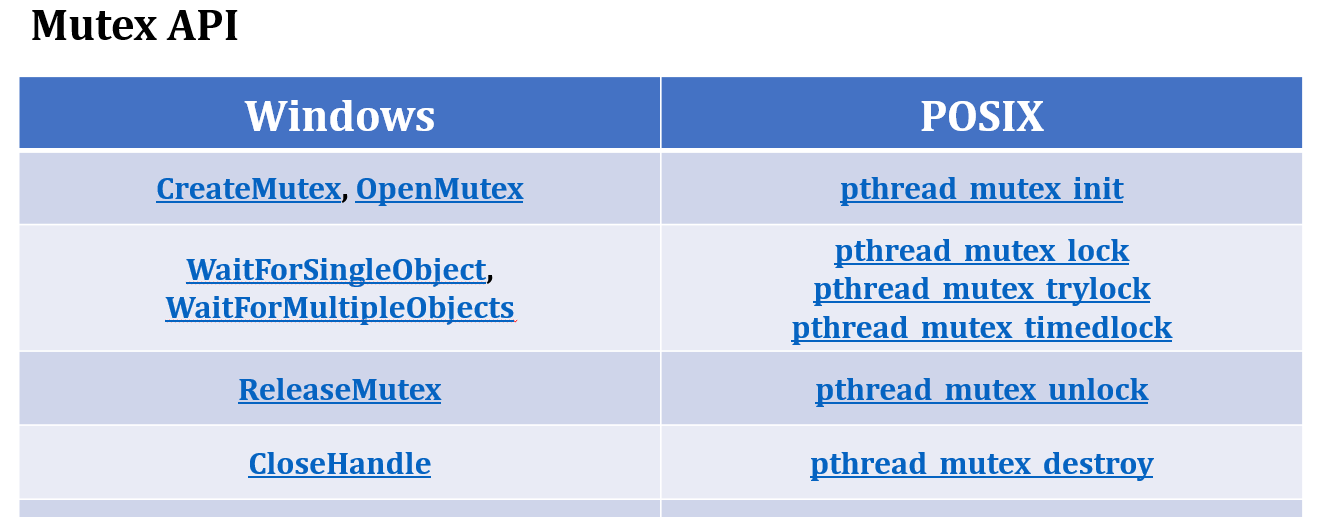
Критическая секция и

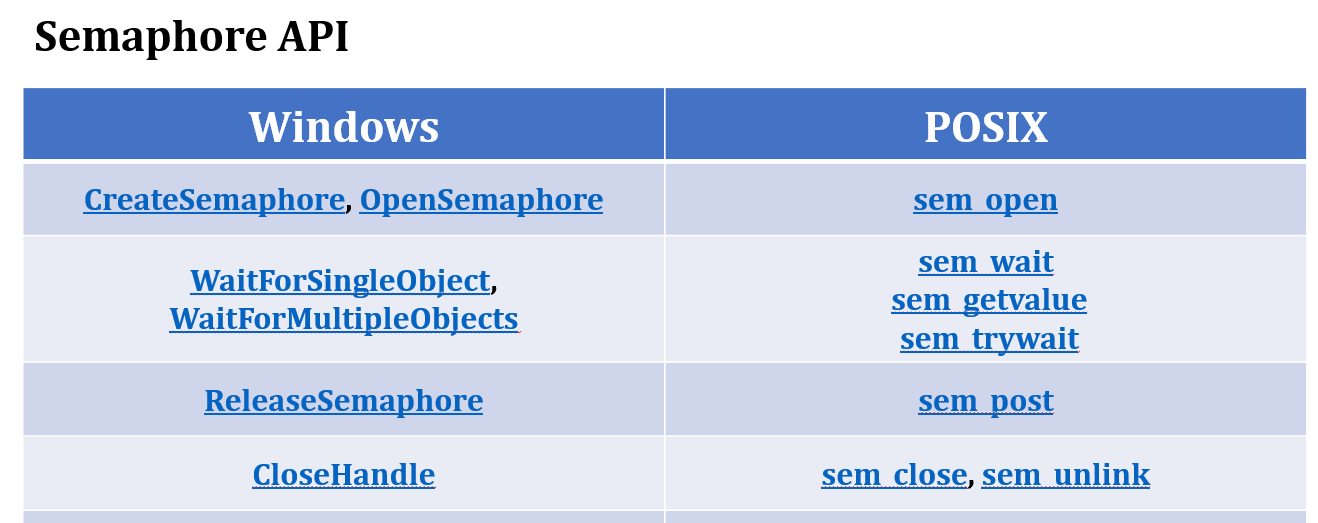


Выбор конкретного механизма зависит от конкретных требований и сценариев использования. Например, если требуется простая и быстрая защита небольших участков кода, критические секции или мьютексы будут оптимальным выбором. Если же необходимо управлять доступом к ресурсу с ограниченной емкостью, лучше использовать семафоры

1. Назовите функции и их назначения, которые предназначены для работы с любым механизмом синхронизации в Windows. В Linux (POSIX).

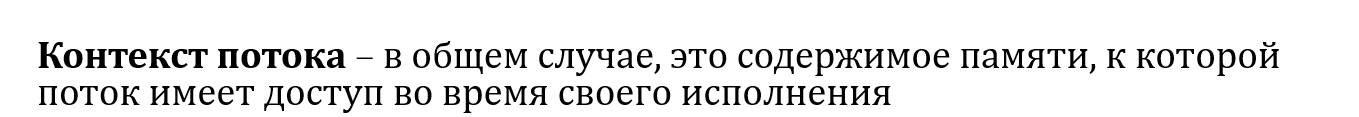




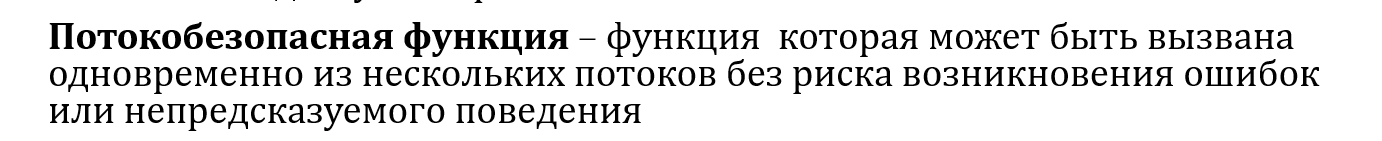


1. Что такое поток? Контекст потока?

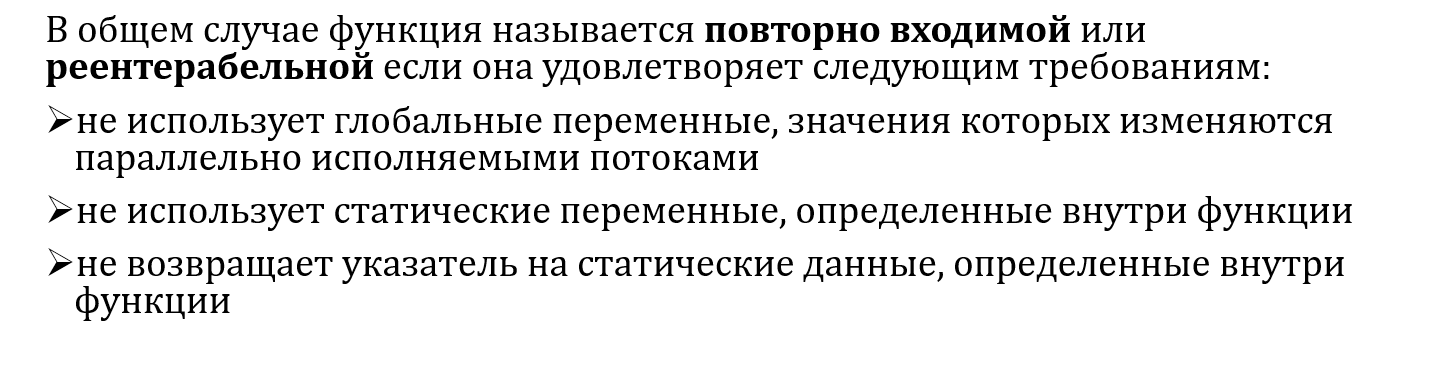




1. Что такое потокобезопасная функция?

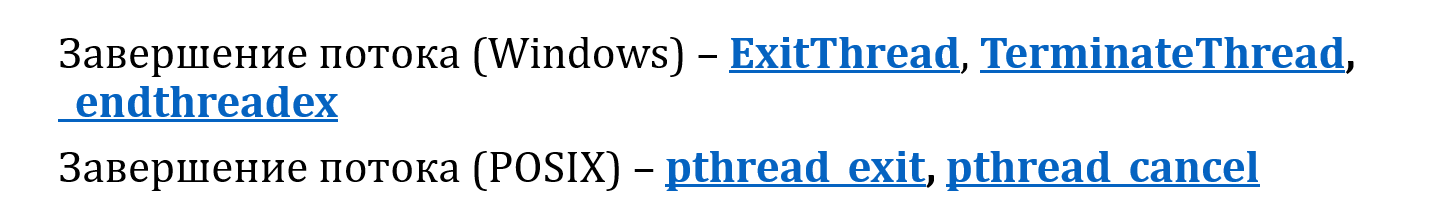


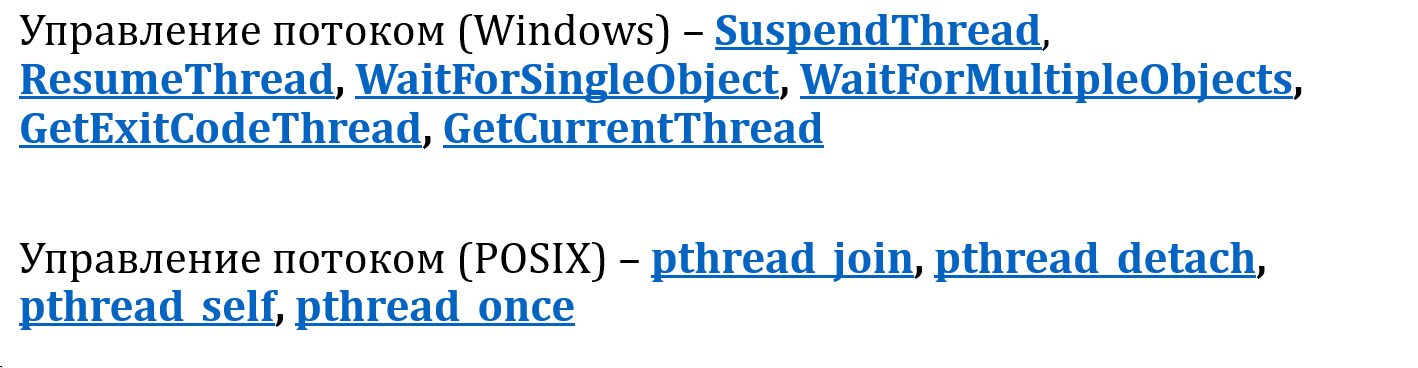
1. Какая функция будет называться реентерабельной?



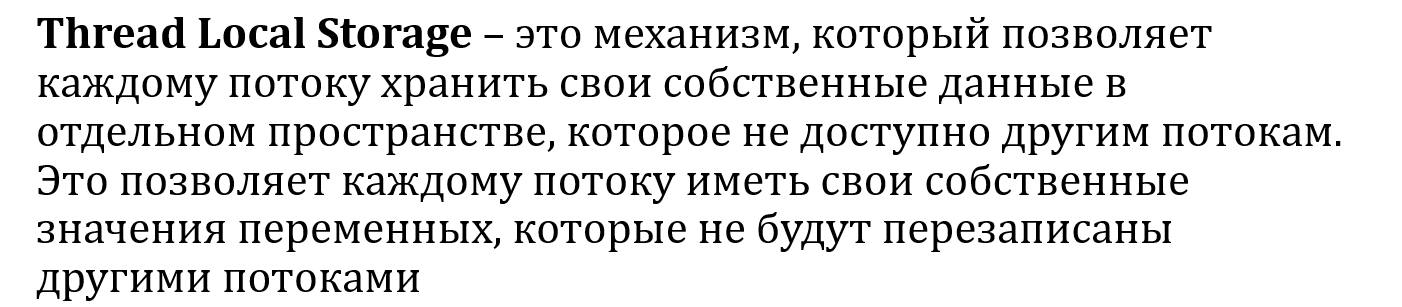
1. Назовите функции и их назначения, которые предназначены для работы с потоками в Windows. В Linux (POSIX).







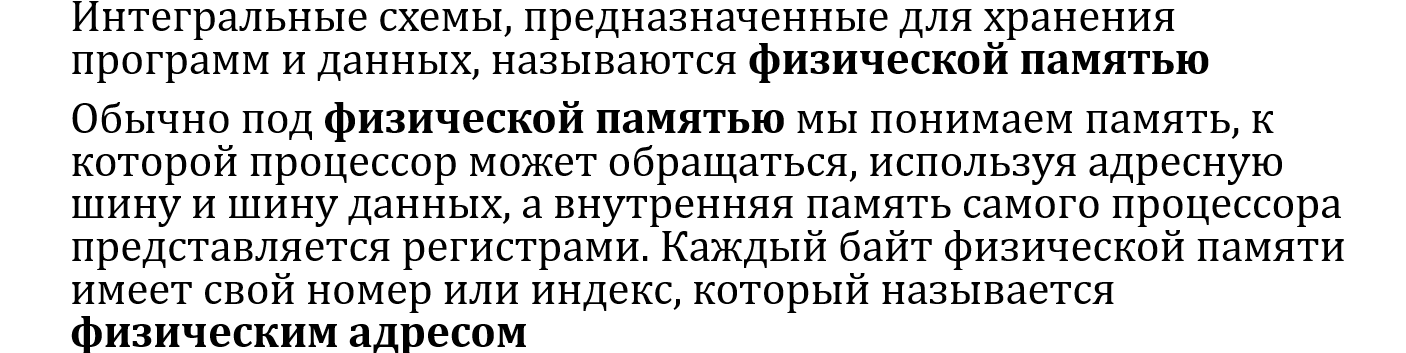
1. Что такое Thread Local Storage?



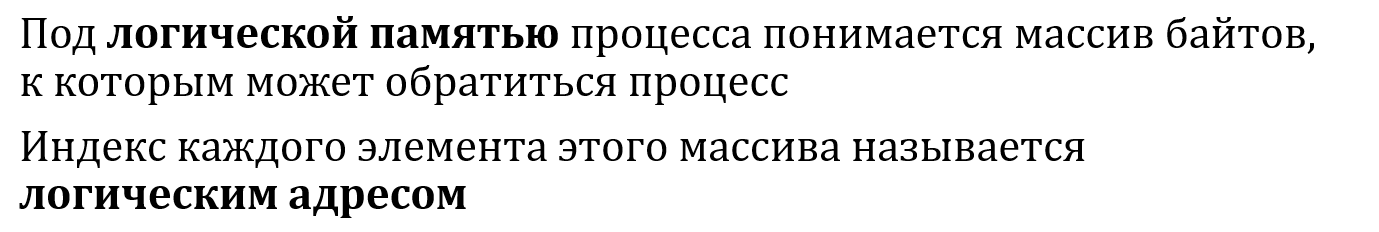
1. В каких случаях лучше использовать многозадачность на основе процессов, а в каких на основе потоков?

Многозадачность на основе процессов лучше использовать, когда важна безопасность и изоляция, так как процессы имеют собственные адресные пространства и не влияют друг на друга. Это также удобно для приложений, использующих разные языки или библиотеки, поскольку процессы могут работать независимо. Контроль ресурсов также проще реализовать на уровне процессов. С другой стороны, многозадачность на основе потоков предпочтительна, когда требуется высокая производительность и частое взаимодействие между компонентами. Потоки легче создавать и обеспечивают общий доступ к памяти, что упрощает обмен данными. Это особенно полезно для приложений, которые должны оставаться отзывчивыми, например, в графических интерфейсах, и для задач, которые могут быть эффективно распараллелены на многоядерных системах.

1. Что такое физическая память?



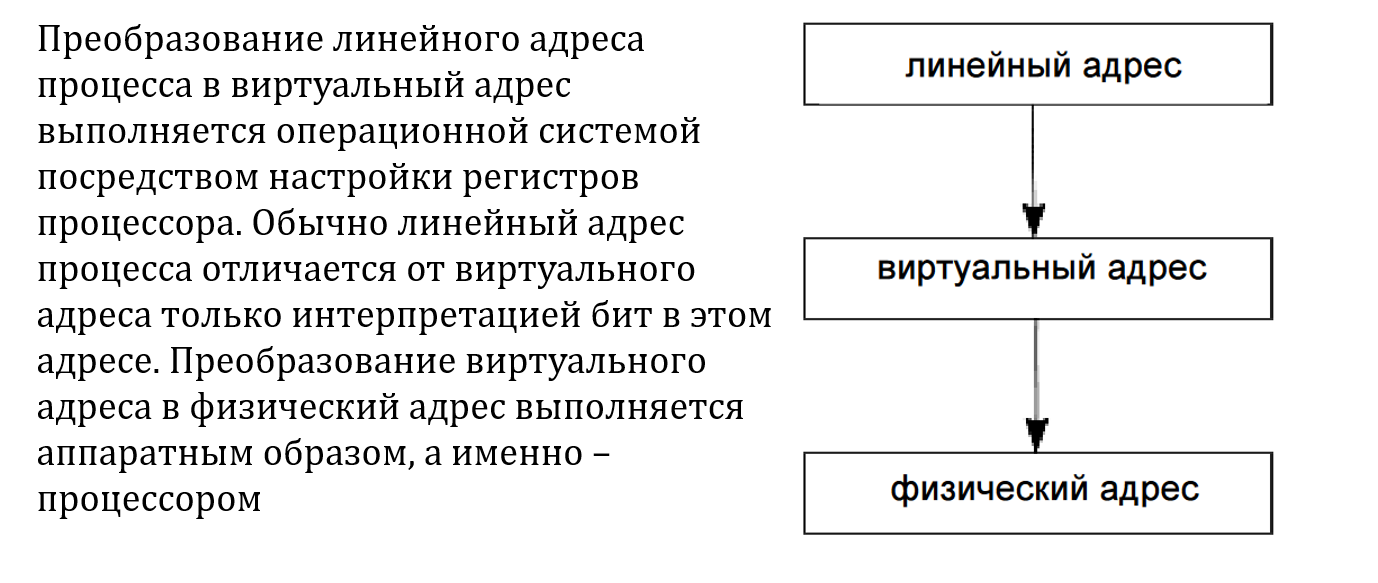
1. Что такое логическая память?



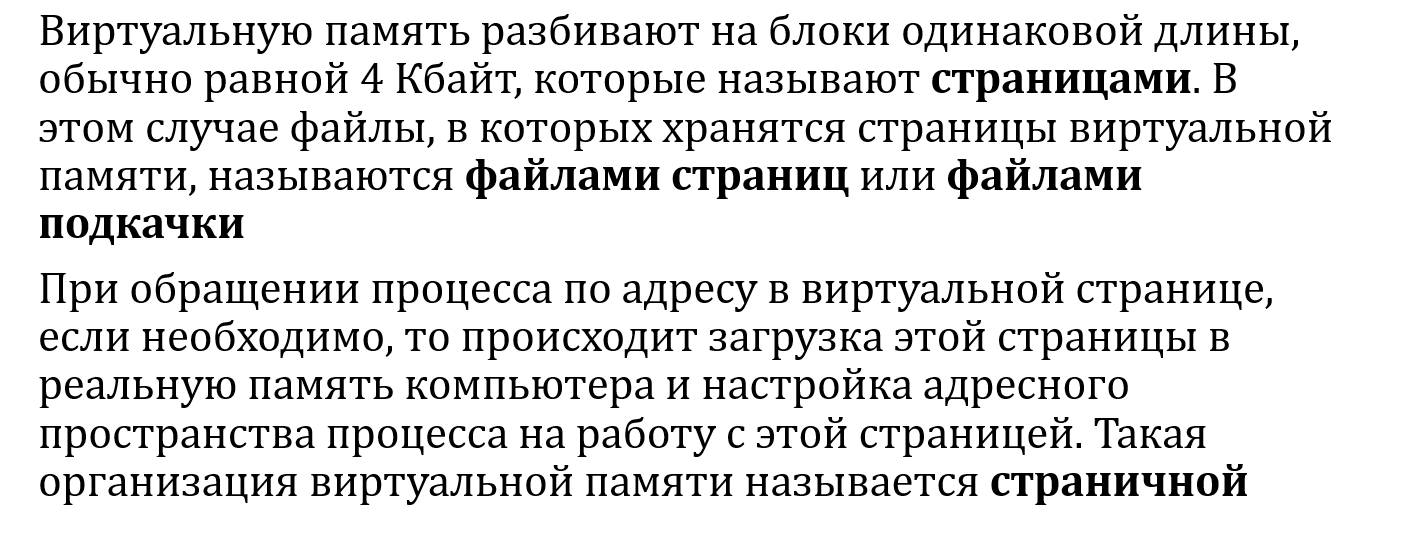
1. Что такое виртуальная память?

Виртуальная память — метод управления памятью компьютера, позволяющий выполнять программы, требующие больше оперативной памяти, чем имеется в компьютере, путём автоматического перемещения частей программы между основной памятью и вторичным хранилищем

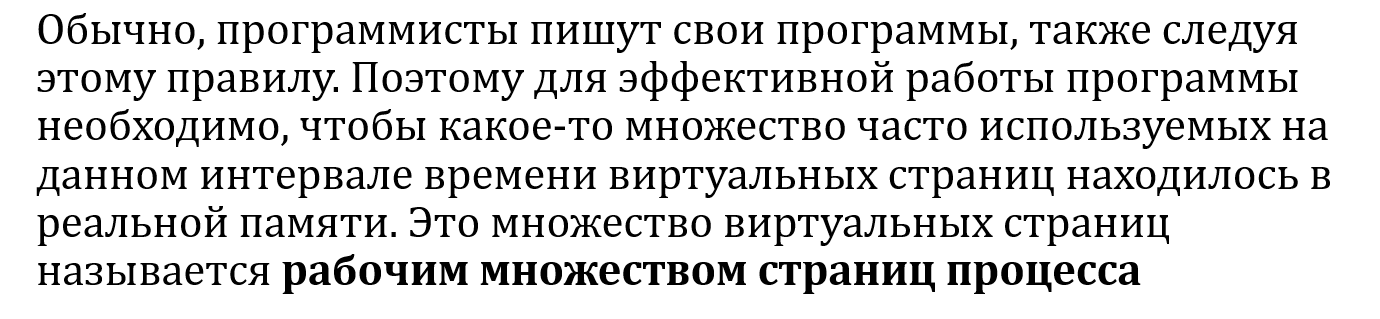
1. Как связаны между собой физический, логический и виртуальный адреса?



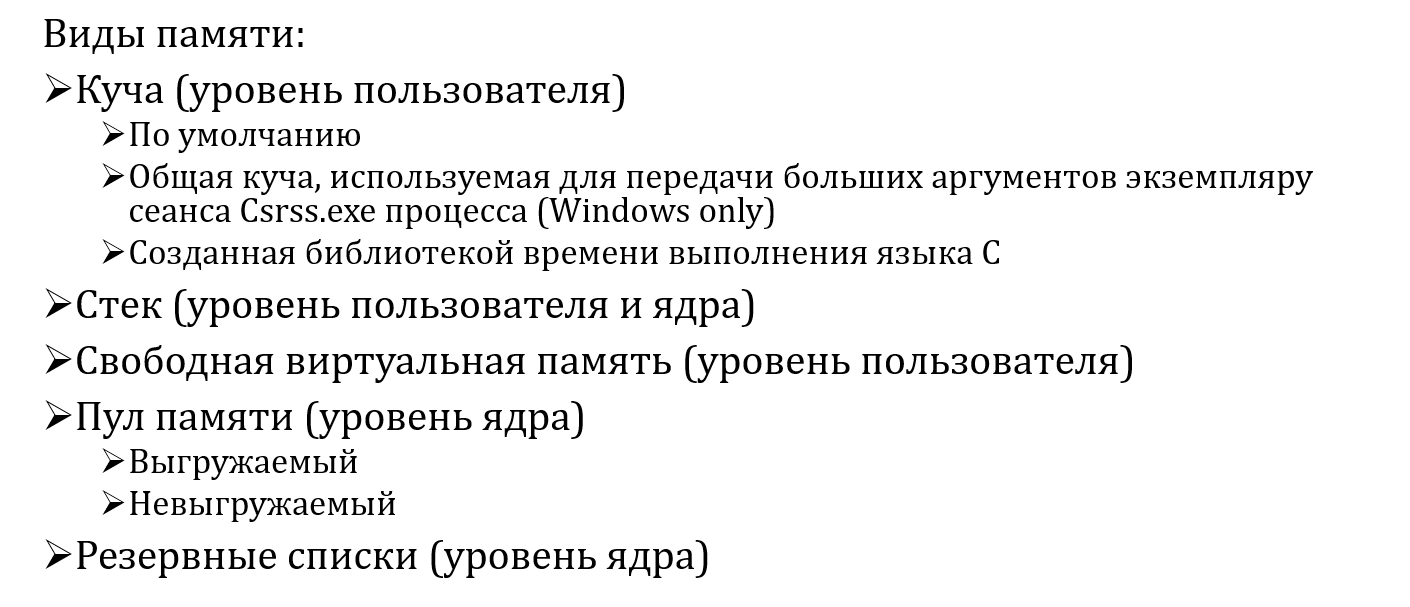
1. Опишите страничную организацию памяти



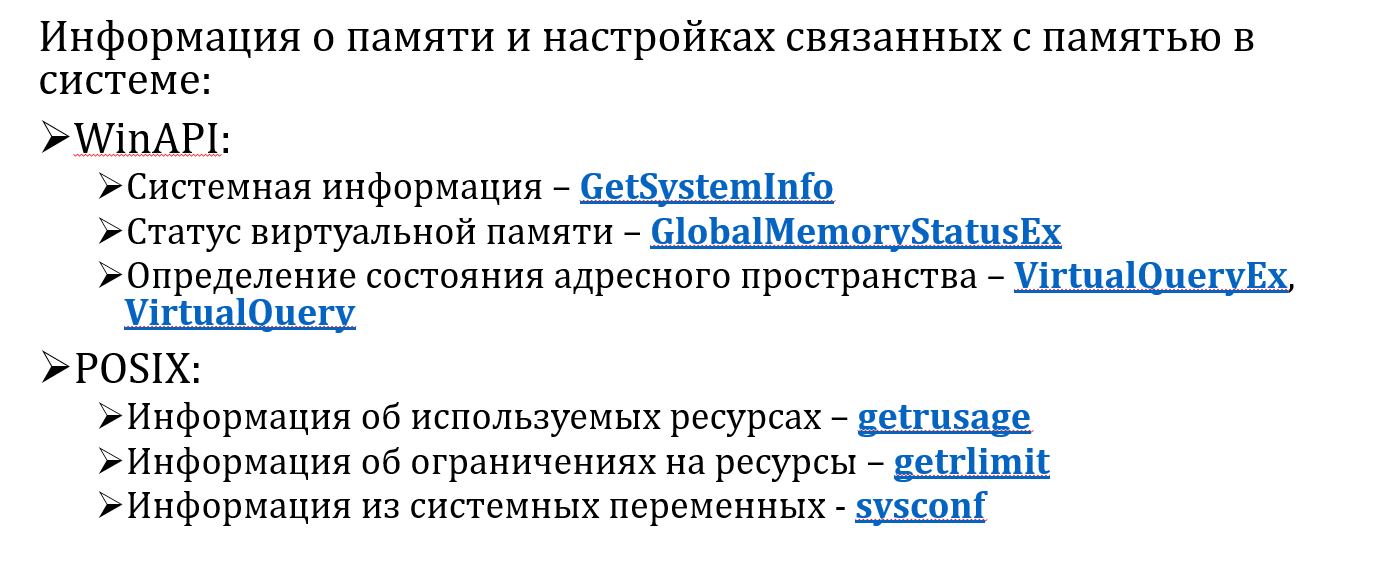
1. Что такое рабочее множество страниц процесса?

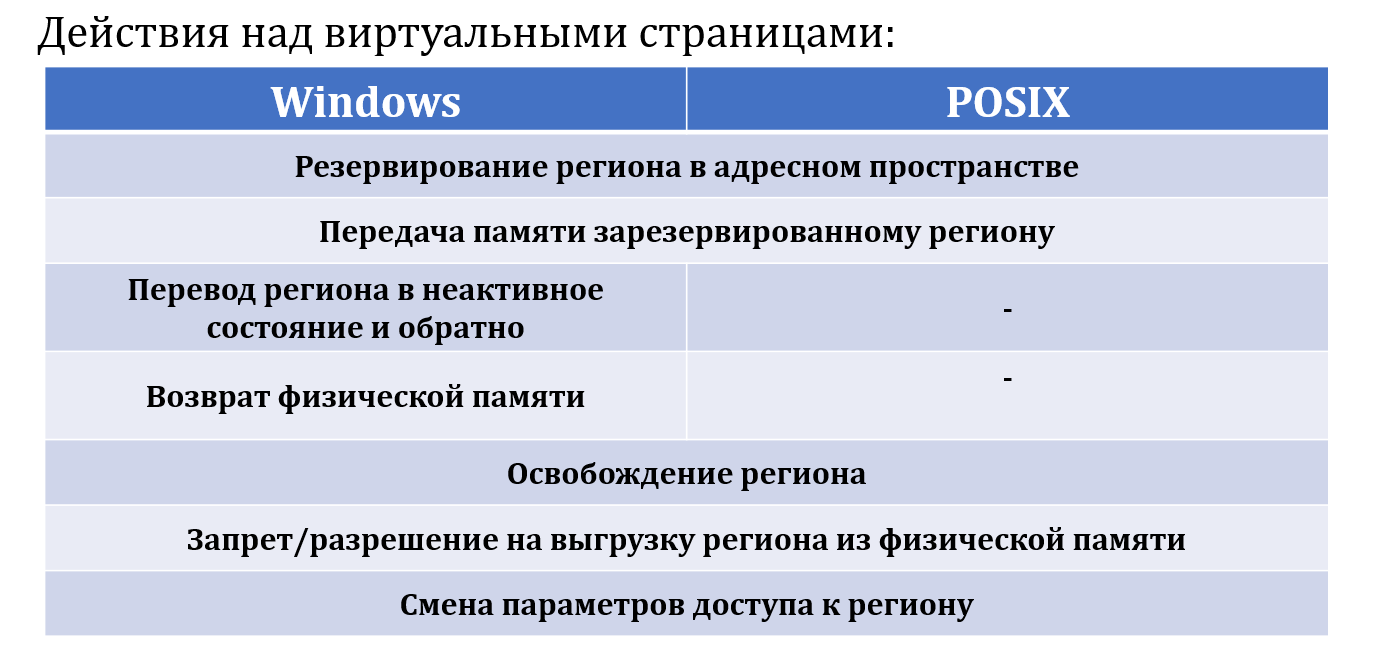


1. Какие виды памяти вам известны?

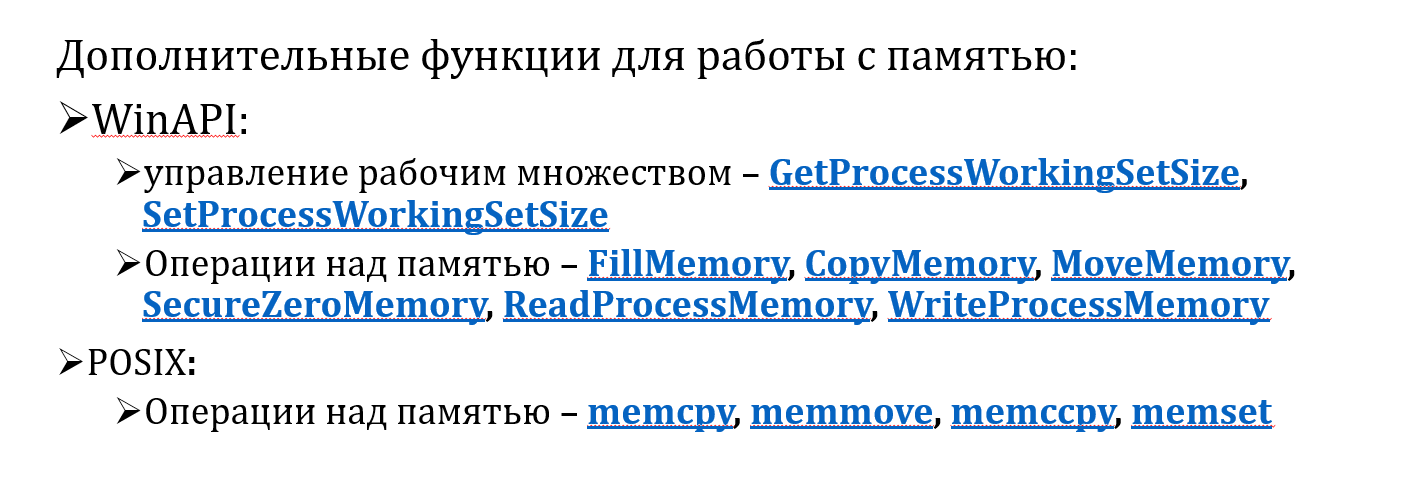


1. Назовите функции и их назначения, которые предназначены для работы с памятью в Windows. В Linux (POSIX).

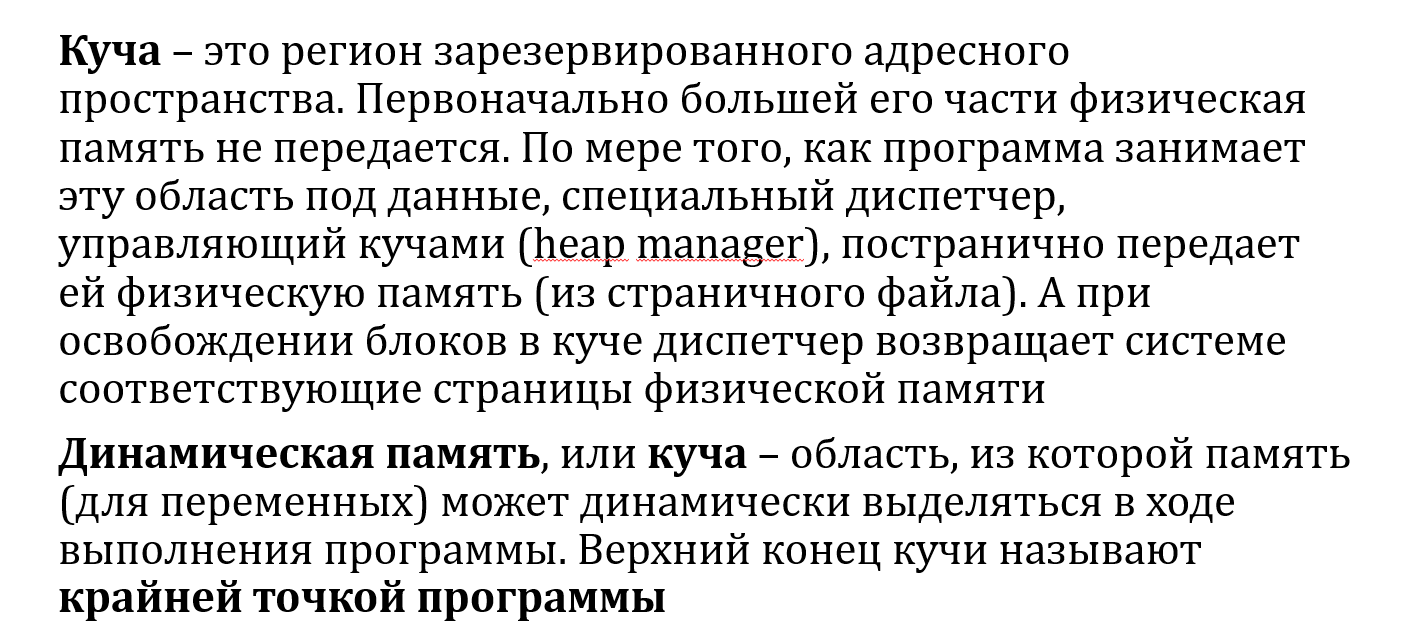


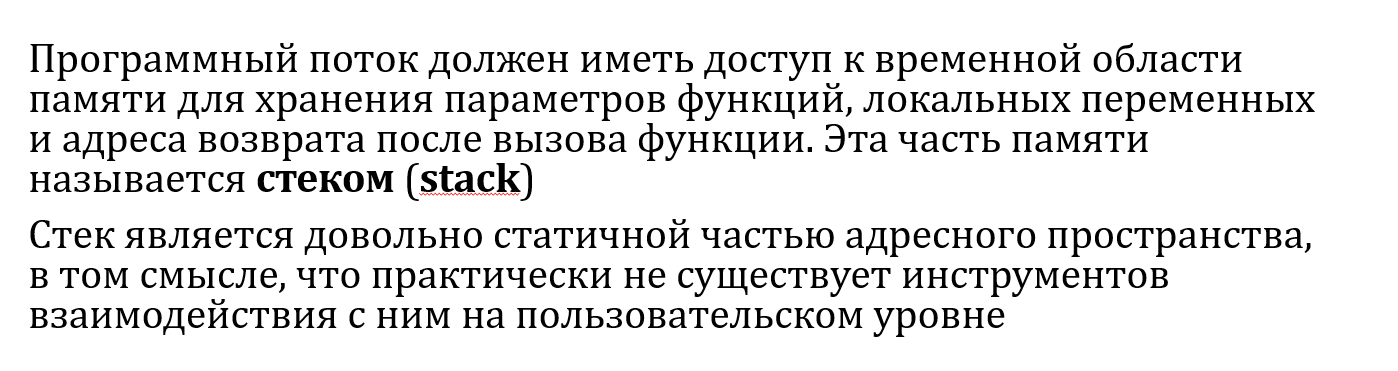




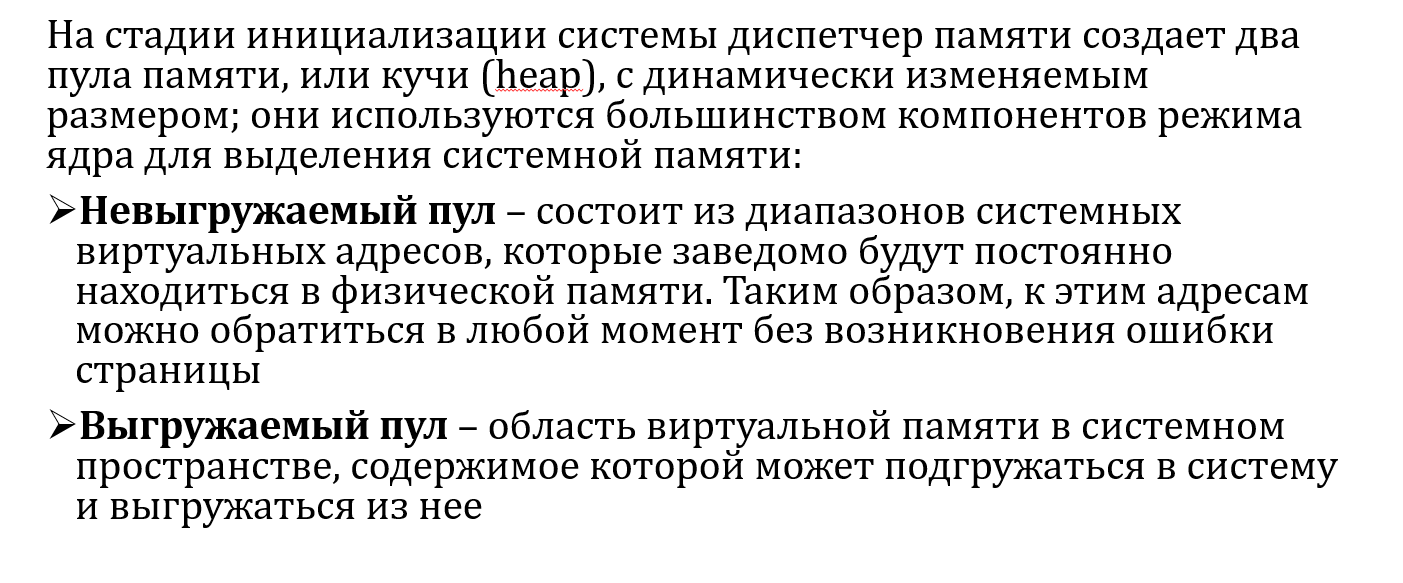


1. Что такое куча и стек?

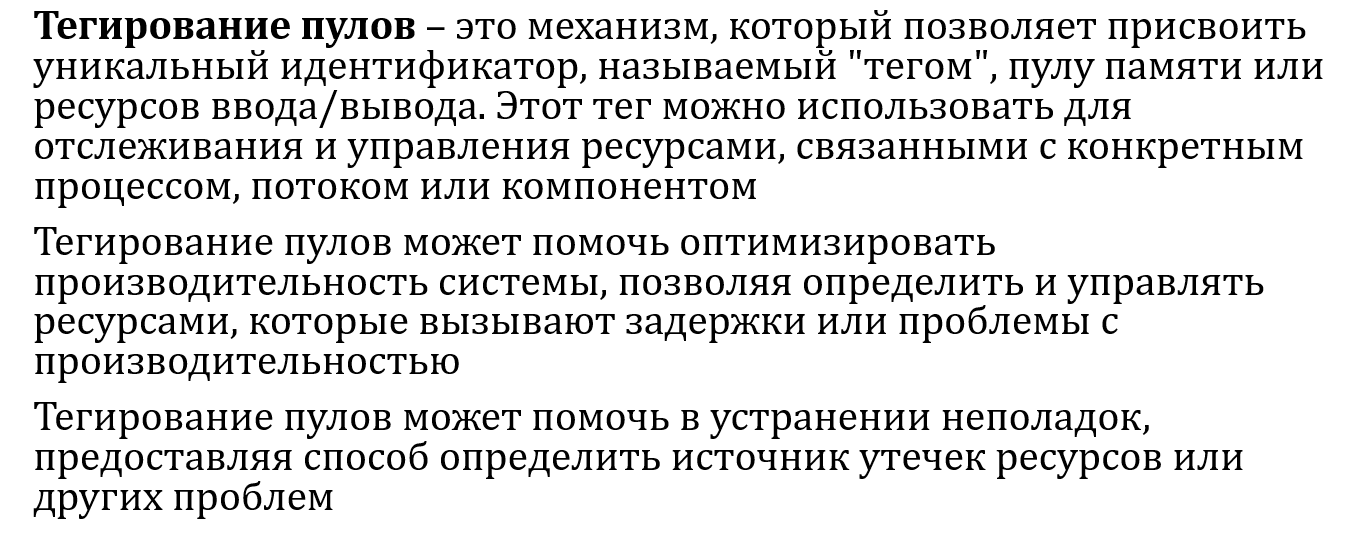




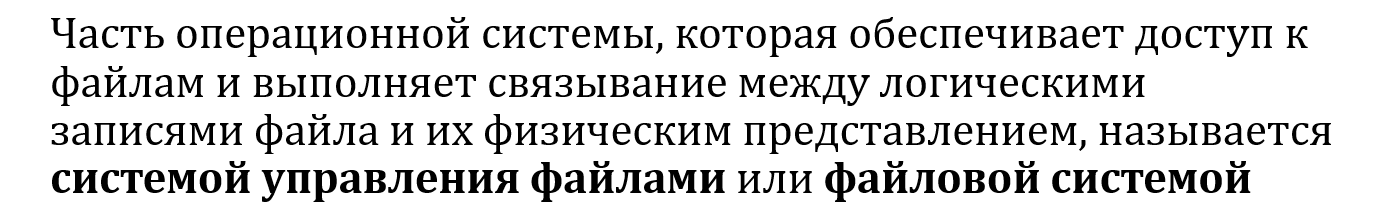
1. Что такое выгружаемый и невыгружаемый пул?



1. Что такое тегирование пула?



1. Что такое файловая система?



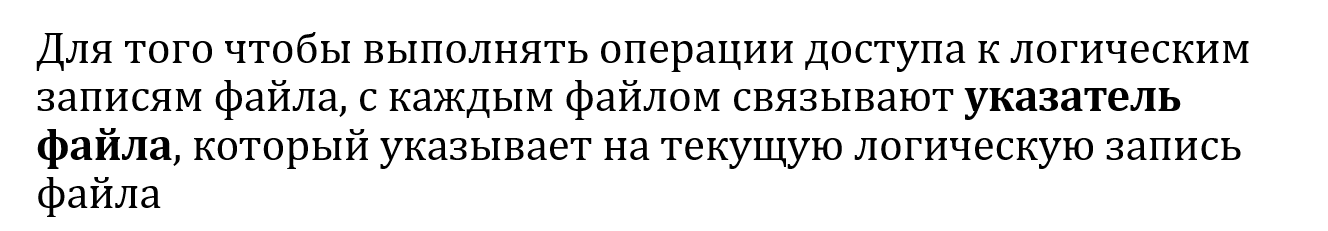
1. Что такое файл? Из чего он состоит?

Файл — это структурированный набор данных, хранящийся на носителе информации (например, жестком диске). Он используется для организации и хранения различных типов информации, таких как текст, изображения или аудио.

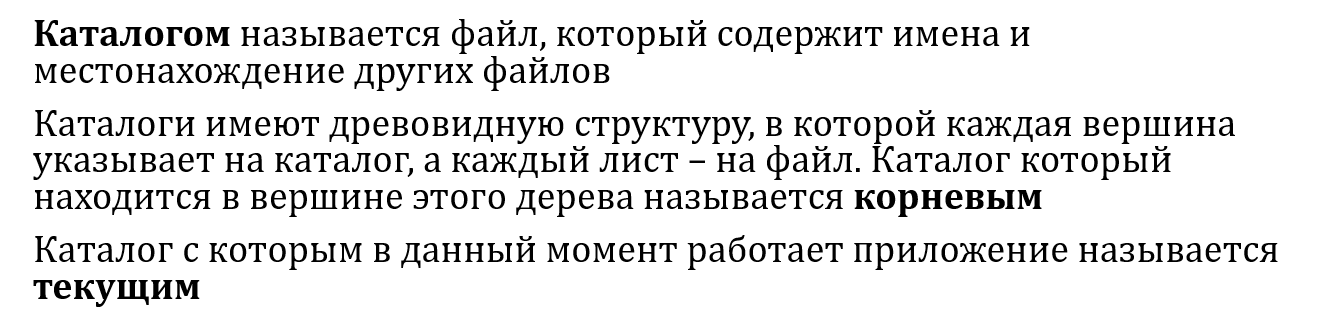
Из чего состоит файл?

* Имя файла: Уникальный идентификатор с расширением (например, .txt, .jpg).
* Метаданные: Информация о файле, включая размер, даты создания и изменения, права доступа.
* Содержимое: Основные данные, хранящиеся в файле (текст, изображения и т.д.).
* Структура: Внутренняя организация данных, зависящая от типа файла (например, заголовки в изображениях).

1. Что такое указатель файла?



1. Что такое каталог?



1. Что такое кеширование?



1. Что такое файл отображенный в память? Как устроена его работа?

Файл, отображённый в память (memory-mapped file), — это способ работы с файлами, который позволяет процессу отображать содержимое файла в адресное пространство своей памяти. Это позволяет программе обращаться к файлу так, будто это обычный массив в оперативной памяти, что упрощает чтение и запись данных.

Как устроена его работа?

Отображение:

При отображении файла в память операционная система создаёт отображение, связывающее файл с определённым диапазоном адресов в памяти. Это делается с помощью системных вызовов, таких как mmap в Unix-подобных системах или CreateFileMapping и MapViewOfFile в Windows.

Доступ к данным:

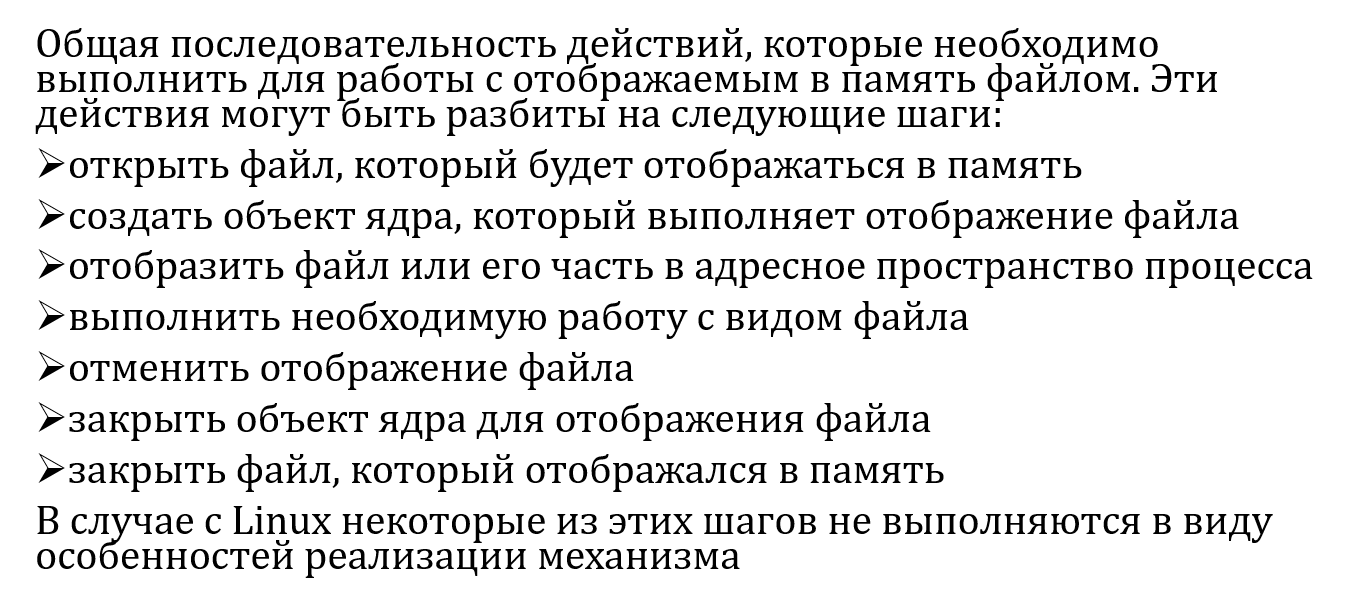
После отображения файл можно читать и записывать, используя указатели в памяти. Это позволяет избежать дополнительных системных вызовов для чтения и записи, так как данные можно обрабатывать как обычные переменные.

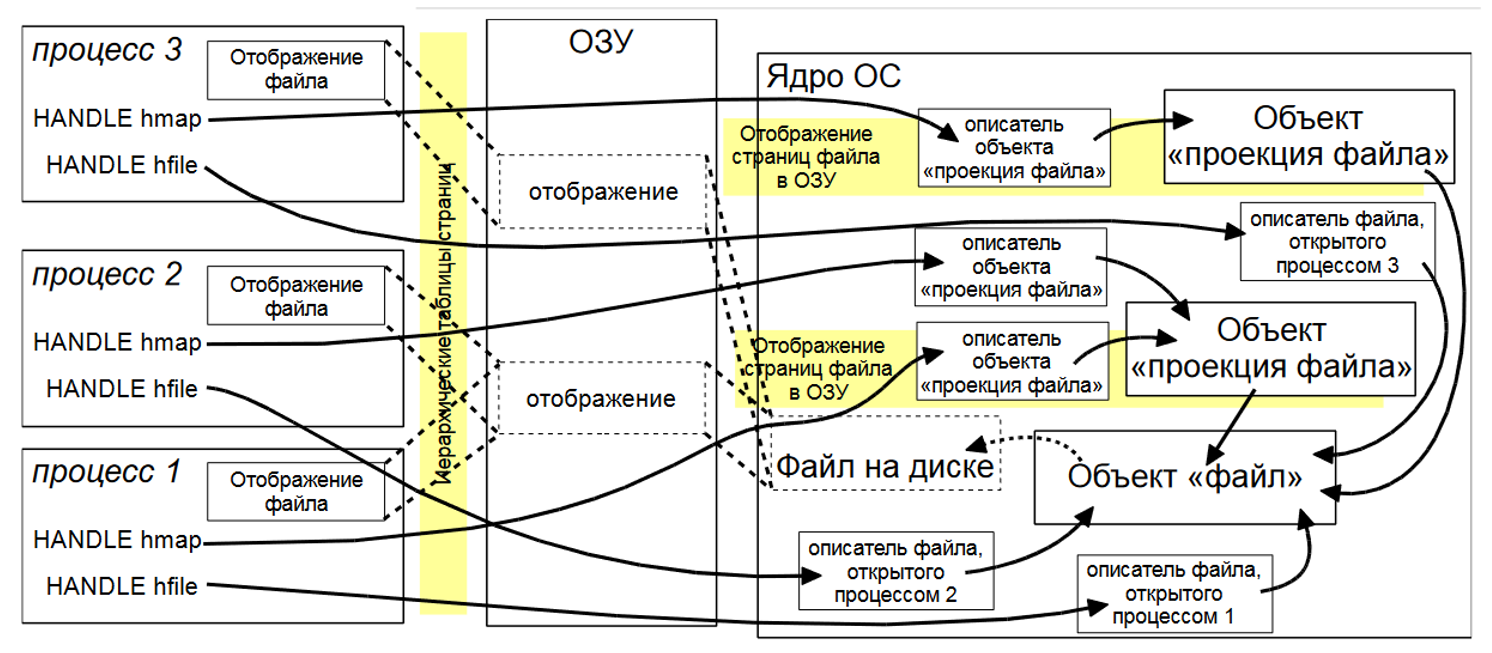
Кэширование:

Операционная система управляет кэшированием данных, что может повышать производительность. Изменения, сделанные в отображаемой области памяти, могут быть автоматически синхронизированы с файлом на диске.

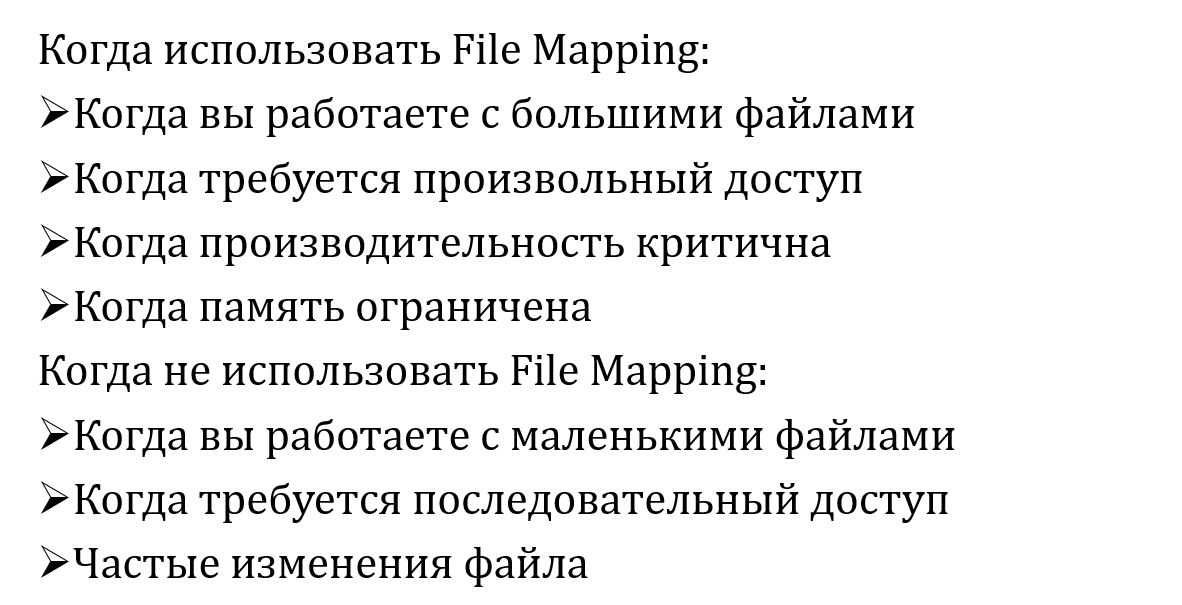
Совместный доступ:

Несколько процессов могут отображать один и тот же файл в память, что позволяет легко делиться данными между ними. Это может быть полезно для реализации межпроцессного взаимодействия.





1. Когда лучше использовать такие файлы?



1. Назовите функции и их назначения, которые предназначены для работы с файлами и каталогами в Windows. В Linux (POSIX).

