**В ПРАВОМ ВЕРХНЕМ УГЛУ**

**Указать: фамилию, инициалы, группу.**

**Вопрос не переписывать, только номер и ответ.**

**ВОПРОСЫ**

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.

Современное деление ПО предусматривает следующие градации:

* Системное ПО
* **Промежуточное (связующее) ПО**
* Прикладное ПО
* Системное программное обеспечение — это набор программ, которые управляют аппаратным обеспечением компьютера и обеспечивают выполнение других программ.
* Промежуточное программное обеспечение, также известное как связывающее ПО (middleware), служит "мостом" между системным ПО и прикладными программами. Оно обеспечивает взаимодействие между различными программами
* Прикладное программное обеспечение — это программы, разработанные для выполнения конкретных задач или решения определенных проблем пользователя. Оно нацелено на конечного пользователя и предоставляет интерфейс для взаимодействия с системой, позволяя пользователям выполнять различные задачи
* **Промежуточное (связующее) ПО** – совокупность программ, осуществляющих управление программными ресурсами, порожденными программами и ориентированными на решение широкого спектра задач
* Например: СУБД, модули управления языком интерфейса ИС, программы сбора и предварительной обработки информации
* **Промежуточное (связующее) ПО** – комплекс технологического ПО для обеспечения взаимодействия между различными приложениями, системами, компонентами
* Например: Веб-сервер, сервер приложений, сервисная шина, система управления контентом

Системное программное обеспечение - это программное обеспечение, которое обеспечивает платформу для другого программного обеспечения.

К функциям системного ПО принято относить:

* 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.)

**Прикладное** программное обеспечение — класс программ, предназначенный для решения практических задач и предназначенный на непосредственное взаимодействие с пользователями. В отличие от системного программного обеспечения, прикладные программы, как правило, не обращаются к ресурсам компьютера напрямую, взаимодействуя с оборудованием и другими программами, используя инфраструктурные и платформные средства — операционные системы, системы управления базами данных, связующее программное обеспечение.

1. Назовите функции системного ПО?

К функциям системного ПО принято относить:

* 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.

**По функциональному назначению**

* **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

**По уровню функциональности**

* 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

1. Что такое системное программирование?

**Системное программирование** – это процесс разработки системного ПО

**Системное программирование** – 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.

**Шаг первый – Предварительная обработка кода**:

* Присоединение исходных файлов
* Работа макропроцессоров

**Шаг второй – Анализ**:

* Лексический анализ

**Шаг второй – Анализ**:

* Синтаксический анализ
* Семантический анализ

**Шаг третий – Синтез**:

* Генерация машинно-независимого кода

**Шаг третий – Синтез**:

* Оптимизация машинно-независимого кода

**Шаг третий – Синтез**:

* Распределение памяти
* Генерация машинного кода
* Оптимизация машинного кода

1. Что такое объектный и загрузочный модули?

**Объектный модуль** – программный модуль, получаемый в результате трансляции исходного модуля

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

1. Что такое операционная система (далее – ОС)?

**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?

* Процессы - абстракция над аппаратными ресурсами процессора.
* Виртуальная память - абстракция над физической памятью компьютера.
* Файлы - абстракция над устройствами ввода-вывода, такими как жесткие диски.

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

1. Каковы задачи ядра ОС?

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?

Пользовательский режим (user mode):

* Режим работы процессора, в котором выполняются пользовательские приложения.
* Приложения в этом режиме имеют ограниченный доступ к аппаратным ресурсам

Режим ядра (kernel mode):

* Режим работы процессора, в котором работает ядро операционной системы.
* Ядро имеет полный доступ ко всем аппаратным ресурсам.
* Ядро управляет критически важными системными компонентами.

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

**Системный вызов** – 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?

Объекты ядра операционной системы - это абстрактные структуры данных и сущности, которые ядро использует для управления различными ресурсами системы.

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 структура данных, используемая операционной системой для хранения информации о ресурсе, таком как процесс, файл или устройство.

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

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

 **Прерывания и возобновления**: Позволяет приостанавливать и восстанавливать выполнение процесса без потери данных.

 **Управления многозадачностью**: Обеспечивает одновременное выполнение нескольких процессов с сохранением их состояний.

 **Изоляции процессов**: Гарантирует, что процессы работают независимо друг от друга.

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).

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)

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

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

 **CreateProcess**:

* Создает новый процесс и его основной поток.

 **TerminateProcess**:

* Завершает указанный процесс.

 **WaitForSingleObject**:

* Ожидает завершения процесса или получения сигнала.

 **GetExitCodeProcess**:

* Получает код завершения указанного процесса.

 **fork**:

* Создает новый процесс (дочерний) путем копирования существующего (родительского).

 **exec**:

* Замещает текущий процесс новым исполняемым файлом.

 **wait**:

* Ожидает завершения дочернего процесса и возвращает его статус.

 **kill**:

* Отправляет сигнал процессу, например, для завершения его работы.

1. Что такое межпроцессное взаимодействие (далее – IPC)?

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

1. Какие категории IPC вам известны?

Категории IPC-механизмов:

* Взаимодействие
* Синхронизация
* Сигналы

1. Какие механизмы входят в каждую категорию?

**1. Взаимодействие:**

* **Каналы (Pipes)**: Однонаправленный или двунаправленный поток данных между процессами.
* **Сообщения (Message Queues)**: Очереди сообщений, позволяющие процессам обмениваться данными.
* **Сокеты (Sockets)**: Механизм для обмена данными между процессами, работающими на разных машинах или в одной системе.

**2. Синхронизация:**

* **Семафоры**: Счетчики, которые управляют доступом к ресурсам.
* **Мьютексы**: Объекты, обеспечивающие взаимное исключение при доступе к общим ресурсам.
* **Условные переменные**: Позволяют процессам ждать определенных условий для продолжения выполнения.

**3. Сигналы:**

* **Сигналы**: Уведомления, отправляемые процессу для информирования о событиях (например, завершение работы).
* **Сигнальные обработчики**: Функции, которые обрабатывают входящие сигналы

1. Опишите общие концепции передачи данных.

Общие концепции:

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

1. Сравните механизмы анонимных и именованных каналов.

**Каналы** бывают **анонимные** и **именованные**

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)

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

**Каналы и очереди:**

* **Когда использовать**: Для простого и структурированного обмена данными между процессами.
* **Примеры**: Асинхронный обмен сообщений, когда важна последовательность и надежность.

**Разделяемая память:**

* **Когда использовать**: Для быстрого доступа к большим объемам данных.
* **Примеры**: Высокопроизводительные вычисления, частый доступ к общим данным.

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

**Windows**

1. **CreatePipe**:Создает анонимный канал для межпроцессного взаимодействия.
2. **WriteFile**:Записывает данные в канал или файл.
3. **ReadFile**:Читает данные из канала или файла.
4. **WaitForSingleObject**:Ожидает завершения объекта синхронизации (например, мьютекса).

**Linux (POSIX)**

1. **pipe**:Создает анонимный канал для межпроцессного взаимодействия.
2. **write**:Записывает данные в канал.
3. **read**:Читает данные из канала.
4. Что такое действие? Контекст действия?

**Действием** называется изменение контекста потока или, другими словами, действием можно назвать любую последовательность команд, которая изменяет контекст потока

Под **контекстом действия** понимается только та часть контекста потока, которая используется этим действием

1. Что такое атомарное действие?

Действие называется **атомарным**, если оно удовлетворяет следующим двум требованиям:

* не прерывается во время своего исполнения
* контекст действия изменяется только самим действием

1. Что такое синхронизация процессов? Потоков?

**синхронизация процессов** – это есть достижение некоторого фиксированного соотношения (порядка) между сигналами, которыми обмениваются эти процессы

Под **синхронизацией потоков** понимается исполнение этими потоками условных непрерывных действий

1. Какие механизмы синхронизации вам известны?

 Семафор:

 Блокировка файла:

 Мьютексы (потоки):

* Условные переменные (потоки)

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

Для синхронизации потоков внутри одного процесса лучше использовать **мьютексы и условные переменные**, так как они:

1. Эффективны: встроенные механизмы ядра, без накладных расходов на переключение контекста.
2. Просты в использовании: имеют понятный API.
3. Обеспечивают согласованный доступ к общим ресурсам.
4. Масштабируются до сотен и тысяч потоков.
5. Назовите функции и их назначения, которые предназначены для работы с любым механизмом синхронизации в Windows. В Linux (POSIX).

**Функции для работы с механизмами синхронизации**

**Windows**

1. **CreateMutex**:Создает мьютекс для синхронизации доступа к ресурсам.
2. **OpenMutex**:
3. **ReleaseMutex**Освобождает мьютекс, разблокируя доступ к ресурсам.
4. **WaitForSingleObject**:Ожидает освобождения объекта синхронизации (например, мьютекса).

**Linux (POSIX)**

1. **pthread\_mutex\_init**:Инициализирует мьютекс.
2. **pthread\_mutex\_lock**Блокирует мьютекс для получения эксклюзивного доступа.
3. **pthread\_mutex\_unlock**:Освобождает мьютекс, разблокируя доступ.
4. Что такое поток? Контекст потока?

**Поток (thread)** - это единица выполнения в пределах процесса.

**Контекст потока** – в общем случае, это содержимое памяти, к которой поток имеет доступ во время своего исполнения

1. Что такое потокобезопасная функция?

**Потокобезопасная функция** – функция которая может быть вызвана одновременно из нескольких потоков без риска возникновения ошибок или непредсказуемого поведения

1. Какая функция будет называться реентерабельной?

**реентерабельной** если она удовлетворяет следующим требованиям:

* не использует глобальные переменные, значения которых изменяются параллельно исполняемыми потоками
* не использует статические переменные, определенные внутри функции
* не возвращает указатель на статические данные, определенные внутри функции

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

Создание потока (Windows) – [**CreateThread**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createthread)(Функция для создания нового потока в Windows.), [**\_beginthreadex**](https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/beginthread-beginthreadex?view=msvc-170)(Функция из библиотеки C, которая создает поток и обеспечивает совместимость с библиотекой C)

Создание потока (POSIX) – [**pthread\_create**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_create.html)(Функция для создания нового потока в POSIX)

Завершение потока (Windows) – [**ExitThread**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-exitthread), [**TerminateThread**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-terminatethread)**,** [**\_endthreadex**](https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/endthread-endthreadex?view=msvc-170)

Завершение потока (POSIX) – [**pthread\_exit**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_exit.html)**,** [**pthread\_cancel**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_cancel.html)

Управление потоком (Windows) – [**SuspendThread**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-suspendthread), [**ResumeThread**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-resumethread)**,** [**WaitForSingleObject**](https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitforsingleobject)**,** [**WaitForMultipleObjects**](https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitformultipleobjects)**,** [**GetExitCodeThread**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-getexitcodethread)**,** [**GetCurrentThread**](https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-getcurrentthread)

Управление потоком (POSIX) – [**pthread\_join**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_join.html)**,** [**pthread\_detach**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_detach.html)**,** [**pthread\_self**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_self.html)**,** [**pthread\_once**](https://pubs.opengroup.org/onlinepubs/9799919799/functions/pthread_once.html)

1. Что такое Thread Local Storage?

**Thread Local Storage** – это механизм, который позволяет каждому потоку хранить свои собственные данные в отдельном пространстве, которое не доступно другим потокам. Это позволяет каждому потоку иметь свои собственные значения переменных, которые не будут перезаписаны другими потоками

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

**Когда использовать многозадачность на основе процессов:**

1. **Изоляция**: Высокая степень изоляции между задачами.
2. **Разные адресные пространства**: Задачи не должны делить память.
3. **Стабильность**: Сбой одного процесса не затрагивает другие.
4. **Разные технологии**: Использование разных языков или технологий.

**Когда использовать многозадачность на основе потоков:**

1. **Производительность**: Быстрое переключение между задачами.
2. **Разделяемая память**: Упрощенный обмен данными между задачами.
3. **Легкость создания**: Быстрое создание и уничтожение задач.
4. **Параллелизм**: Эффективное выполнение параллельных задач.

Когда лучше использовать поточную многозадачность:

* Ограниченность системных ресурсов
* Частое использование данных между подзадачами
* Частая синхронизация подзадач
* Поддержка языками и библиотеками
* Задачи с упором на вычисления (так называемые CPU-bound tasks)
* Написание GUI
* Задачи с требованием к масштабированию

1. Что такое физическая память?

Интегральные схемы, предназначенные для хранения программ и данных, называются **физической памятью**

1. Что такое логическая память?

Под **логической памятью** процесса понимается массив байтов, к которым может обратиться процесс

1. Что такое виртуальная память?

Для решения этой задачи физическую память компьютера дополняют памятью на дисках. Полученную расширенную память называют **виртуальной памятью**

**Виртуальная память**:

* Это концепция, позволяющая программе использовать больше памяти, чем физически доступно в системе. Виртуальная память объединяет физическую и логическую память, предоставляя каждому процессу его собственное адресное пространство, которое может быть больше, чем доступная физическая память. Виртуальная память также включает механизм, который позволяет загружать данные из дискового пространства (например, жесткого диска или SSD) в RAM по мере необходимости

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

 **Преобразование**:

* Логические и виртуальные адреса преобразуются в физические адреса с помощью механизма управления памятью (например, таблиц страниц).

 **Изоляция**:

* Виртуальные адреса позволяют процессам работать независимо друг от друга, создавая иллюзию отдельного адресного пространства, в то время как физические адреса относятся к реальной памяти.

1. Опишите страничную организацию памяти.

Виртуальную память разбивают на блоки одинаковой длины, обычно равной 4 Кбайт, которые называют **страницами**. В этом случае файлы, в которых хранятся страницы виртуальной памяти, называются **файлами страниц** или **файлами подкачки**

При обращении процесса по адресу в виртуальной странице, если необходимо, то происходит загрузка этой страницы в реальную память компьютера и настройка адресного пространства процесса на работу с этой страницей. Такая организация виртуальной памяти называется **страничной**

1. Что такое рабочее множество страниц процесса?

Обычно, программисты пишут свои программы, также следуя этому правилу. Поэтому для эффективной работы программы необходимо, чтобы какое-то множество часто используемых на данном интервале времени виртуальных страниц находилось в реальной памяти. Это множество виртуальных страниц называется **рабочим множеством страниц процесса**

1. Какие виды памяти вам известны?

Виды памяти:

* Куча (уровень пользователя)
  + По умолчанию
  + Общая куча, используемая для передачи больших аргументов экземпляру сеанса Csrss.exe процесса (Windows only)
  + Созданная библиотекой времени выполнения языка С
* Стек (уровень пользователя и ядра)
* Свободная виртуальная память (уровень пользователя)
* Пул памяти (уровень ядра)
  + Выгружаемый
  + Невыгружаемый
* Резервные списки (уровень ядра)

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

Windows API содержит четыре группы функций для управления памятью в приложениях:

* [API Virtual](https://learn.microsoft.com/en-us/windows/win32/memory/memory-management-functions)
* [API Heap](https://learn.microsoft.com/en-us/windows/win32/memory/memory-management-functions)
* [Локальные/глобальные API](https://learn.microsoft.com/en-us/windows/win32/memory/memory-management-functions) (в целях совместимости с 16-битными приложениями)
* [Файлы, отображенные в память](https://learn.microsoft.com/en-us/windows/win32/memory/memory-management-functions)

POSIX содержат такие группы функций для управления памятью в приложениях как:

* [API mman](https://pubs.opengroup.org/onlinepubs/9799919799/basedefs/sys_mman.h.html)
* [API heap/stack](https://pubs.opengroup.org/onlinepubs/9799919799/basedefs/stdlib.h.html) (аналогично стандартной библиотеке С)
* Файлы, отображенные в память (mman)

1. Что такое куча и стек?

**Куча** — это область динамической памяти, используемая для хранения данных, которые могут изменяться по размеру и времени жизни во время выполнения программы.

Программный поток должен иметь доступ к временной области памяти для хранения параметров функций, локальных переменных и адреса возврата после вызова функции. Эта часть памяти называется **стеком** (**stack**)

1. Что такое выгружаемый и невыгружаемый пул?

* **Выгружаемый пул** – область виртуальной памяти в системном пространстве, содержимое которой может подгружаться в систему и выгружаться из нее
* **Невыгружаемый пул** – состоит из диапазонов системных виртуальных адресов, которые заведомо будут постоянно находиться в физической памяти. Таким образом, к этим адресам можно обратиться в любой момент без возникновения ошибки страницы

1. Что такое тегирование пула?

**Тегирование пулов** – это механизм, который позволяет присвоить уникальный идентификатор, называемый "тегом", пулу памяти или ресурсов ввода/вывода. Этот тег можно использовать для отслеживания и управления ресурсами, связанными с конкретным процессом, потоком или компонентом

1. Что такое файловая система?

Часть операционной системы, которая обеспечивает доступ к файлам и выполняет связывание между логическими записями файла и их физическим представлением, называется **системой управления файлами** или **файловой системой**

1. Что такое файл? Из чего он состоит?

**Файл** — это структурированная единица хранения данных на носителе информации, которая может быть прочитана и записана операционной системой или приложениями.

 **Заголовок** (не всегда присутствует):.

 **Данные**:

 **Конец файла**:

1. Что такое указатель файла?

Для того чтобы выполнять операции доступа к логическим записям файла, с каждым файлом связывают **указатель файла**, который указывает на текущую логическую запись файла

1. Что такое каталог?

**Каталогом** называется файл, который содержит имена и местонахождение других файлов

1. Что такое кеширование?

**Кэширование** – процесс записи данных в память с более быстрым доступом

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

**Файл, отображённый в память** (memory-mapped file) — это механизм, который позволяет отображать содержимое файла в адресное пространство процесса, что позволяет работать с файлом как с обычным массивом памяти.

 **Отображение**:

* Файл отображается в память с помощью системных вызовов, таких как mmap в POSIX или CreateFileMapping и MapViewOfFile в Windows.

 **Доступ к данным**:

* После отображения файл может быть доступен через указатели, как если бы это была обычная память. Это упрощает работу с данными и позволяет использовать стандартные операции чтения и записи.

 **Синхронизация**:

* Изменения, внесённые в отображённые данные, могут быть автоматически синхронизированы с файлом на диске. Это может происходить либо немедленно, либо при завершении работы с отображением.

 **Производительность**:

* Использование отображенных файлов может повысить производительность за счёт уменьшения количества операций ввода-вывода. Доступ к данным в памяти обычно быстрее, чем доступ к данным на диске.

 **Совместное использование**:

* Отображенные файлы могут быть использованы для межпроцессного взаимодействия, позволяя нескольким процессам работать с одним и тем же файлом в памяти.

1. Когда лучше использовать такие файлы?

 **Большие файлы**: Для работы с файлами, которые не помещаются в оперативную память.

 **Частые операции**: При необходимости многократного чтения и записи данных.

 **Параллельная работа**: Для межпроцессного взаимодействия с общими файлами.

 **Упрощение кода**: Для более простого доступа к данным как к массивам.

 **Динамические изменения**: Когда данные часто изменяются и требуется автоматическая синхронизация.

1. Назовите функции и их назначения, которые предназначены для работы с файлами и каталогами в Windows. В Linux (POSIX).
2. **CreateFile**:
   * Создает или открывает файл или устройство.
3. **ReadFile**:
   * Читает данные из открытого файла.
4. **WriteFile**:
   * Записывает данные в открытый файл.
5. **CloseHandle**:
   * Закрывает дескриптор файла.
6. SetFilePointerEx:
7. GetFileAttributes:
8. SetFileAttributes:
9. LockFile
10. UnlockFile

**Linux (POSIX)**

1. **open**:
   * Открывает файл или создает его.
2. **read**:
   * Читает данные из файла.
3. **write**:
   * Записывает данные в файл.
4. **close**:
   * Закрывает дескриптор файла.
5. **unlink**:
   * Удаляет файл.
6. **mkdir**:
   * Создает новый каталог.
7. **rmdir**:
   * Удаляет каталог.
8. **stat**:
   * Получает информацию о файле, включая атрибуты.