# +1. Что такое POSIX?

набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов. Стандарт создан для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости прикладных программ на уровне исходного кода, но может быть использован и для не-Unix систем

# +2. Что такое системный вызов?

Системный вызов (System Call) - обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции

# +3. Что такое аппаратное прерывание, программное прерывание?

**Аппаратное** - возникает как реакция микропроцессоров на физический сигнал от некоторого устройства (клавиатура, системные часы, жесткий диск и т.д.), по времени возникновения эти прерывания асинхронны, т.е. происходят в случайные моменты времени

**Программное** - вызываются искусственно с помощью соответствующей команды из программы, предназначены для выполнения некоторых действий операционной системы, являются синхронными;

# +4. Что такое процесс?

Процесс OS – это экземпляр выполняемой программы в операционной системе. Каждый процесс имеет свою собственную память, стек вызовов и ресурсы, такие как файлы и сетевые соединения

# +5. Что такое контекст процесса?

состояние процесса, включая значения регистров, содержимое памяти, **Идентификатор процесса (PID), стек вызовов** и другие ресурсы, необходимые для его выполнения. Контекст процесса позволяет операционной системе переключать выполнение между разными процессами.

# 6. Что такое родительский и дочерний процесс?

Родительский процесс – это процесс, который создает другой процесс, называемый дочерним.

# 7. Что такое процесс инициализации OS?

Процесс инициализации запускается первым при загрузке операционной системы и запускает все другие процессы, которые должны быть запущены, это родительский процесс для всего, что выполняется в системе.

Его главная задача - инициализация и подготовка операционной системы к работе, включая запуск других процессов и основных системных служб

# +8. Перечислите области памяти процесса и поясните их назначение.

code, static, heap, data, stack

-----Возрастание адресов----->

* Сегмент кода (Code Segment): Содержит исполняемый код приложения.
* Сегмент данных (Data Segment): Содержит переменные и глобальные данные.
* Сегмент стека (Stack Segment): Используется для хранения вызовов функций и локальных переменных.
* Сегмент кучи (Heap Segment): Место для динамического выделения памяти.
* **Static – статические переменные**

# 9. Чем отличаются системные процессы от пользовательских?

**системные** **процессы:** процессы запускаемые автоматически при запуске OS, выполняют системные задачи и служат для обеспечения функционирования операционной системы и управления ей, в то время как **пользовательские процессы** выполняют задачи, связанные с прикладными программами.

Системные процессы имеют доступ к системным ресурсам и операциям, таким как управление железом и файловой системой, чего не имеют пользовательские процессы.

Системные процессы являются частью ядра и всегда находятся в оперативной памяти. Такие процессы не имеют соответствующих им программ в виде исполняемых файлов и запускаются особым образом при инициализации ядра системы

# 10. Что такое Windows-сервисы, Linux-демоны?

**Windows-сервисы** - приложения, автоматически выполняемое системой при запуске операционной системы Windows и выполняющиеся вне зависимости от статуса пользователя. работают в фоновом режиме и выполняют системные задачи

**Linux-демоны** – это служебные программы, которые работают в фоновом режиме, выполняют различные системные задачи и служат для обеспечения работы ОС

Или

**Windows-сервисы** – это фоновые приложения, которые выполняются в операционной системе Windows и предназначены для выполнения различных системных задач и служб. Эти службы могут выполняться автоматически при запуске системы и работать в фоновом режиме, не требуя активного взаимодействия с пользователем.

# +11. С помощью каких системных вызовов можно создать дочерний процесс в Windows? Поясните разницу.

CreateProcess - данный вызов имеет множество параметров, многие из которых «по умолчанию» Функция CreateProcess создает новый процесс и его первичный (главный) поток

# +12. С помощью каких системных вызовов можно создать дочерний процесс в Linux? Поясните разницу.

fork, system

**fork()** создает точную копию вызывающего процесса. После выполнения системного вызова fork два процесса, родительский и дочерний, имеют единый образ памяти, единые строки описания конфигурации и одни и те же открытые файлы

Дочерний процесс полностью копирует память и состояние родительского процесса, включая код, данные, открытые файловые дескрипторы, переменные и ресурсы. Дочерний процесс начинает выполнение с той же инструкции, с которой был вызван **fork()** в родительском процессе. В дальнейшем родительский и дочерний процессы могут выполняться параллельно и независимо друг от друга. Изменения в памяти или состоянии одного процесса не влияют на другой.

**exec()** не создает новый процесс, а заменяет данные текущего процесса новыми данными. Вызов exec происходит таким образом, что переданная в качестве аргумента программа загружается в память вместо старой, которая вызвала exec.

**system()** создает новый процесс, который затем выполняет команду, переданную в качестве аргумента. Процесс, вызвавший команду system, ожидает завершения выполнения команды в новом процессе.

Ваша программа передает команду, содержащуюся в **command**, в командную оболочку текущей операционной системы Командная оболочка запускает новый процесс и выполняет указанную команду.

неявно запускает новый процесс для обслуживания запроса и возвращает значение, полученное через дочерний процесс, который он изначально разветвил.

# +13. Какие потоки данных доступны любому процессу автоматически?

ввода, вывода, вывод ошибок

# 14. Поясните назначение системного вызова WaitForSingleObject в Windows-приложении.

для ожидания завершения выполнения дочерних процессов или потоков. Приостанавливает выполнение потока (или процесса) до тех пор, пока не произойдет одно из двух событий:

* истечет таймаут ожидания;
* выполнение указанного объекта синхронизации (ожидаемый объект перейдет в сигнальное (signaled) состояние.)

# 15. Поясните назначение системного вызова wait в Linux-приложении.

используется в родительских процессах для ожидания завершения выполнения и получения статуса завершения дочерних процессов.

# 16. Дайте развернутое определение процесса OS.

Процесс в операционной системе (OS) – это исполняемая программа или задача, которая выполняется в оперативной памяти компьютера. Он представляет собой изолированную и независимую единицу выполнения, обладающую своим собственным набором ресурсов и состоянием.

Каждый процесс имеет свое собственное **адресное пространство** **памяти**, что обеспечивает изоляцию данных и кода одного процесса от других процессов. -

для обмена данными между процессами, применяется специальный механизм – **механизм межпроцессного взаимодействия (IPC);**

Каждый процесс имеет доступ к своим собственным **ресурсам,** таким как файлы, открытые дескрипторы, память, регистры CPU и другие

**Контекст выполнения**

Каждый процесс в ОС имеет уникальный **идентификатор процесса (PID),** который используется для идентификации и управления процессом.

процесс OS – единица работы OS - объект ядра OS+адресное пространство:

* **создается ядром OS по системному вызову;**
* адресное пространство (данные, программа, стек, куча);
* ресурсы: регистры, открытые файлы, родительский процесс, перечень связанных (дочерних) процессов, реальные страницы памяти, виртуальное адресное пространство, маркеры доступа (безопасность);
* **процесс может создавать (с помощью системного вызова) дочерние процессы,** в общем случае может образовываться дерево процессов;чя
* обычно соответствует работающей программе (например, exe-файлу в Windows);
* OS хранит список (или таблицу) объектов работающих процессов;
* **при приостановке процесса в объекте процесса сохраняется вся информация** (регистры, уведомления OS,… - контекст процесса), позволяющая возобновить работу процесса;
* процессы изолированы друг от друга;
* для обмена данными между процессами, применяется специальный механизм – механизм межпроцессного взаимодействия (IPC);

Дискрипторы

<https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/handles-and-objects>

*указатель на область памяти который использует ядро ос для взаимодействии с объектом*

***Дескрипторы и объекты***

*Объект* — это структура данных, представляющая системный ресурс, например файл, поток или графическое изображение. Приложение не может напрямую обращаться к данным объекта, а также к системным ресурсам, которые представляет объект. Вместо этого приложение должно получить *дескриптор* объекта, который он может использовать для проверки или изменения системного ресурса. Каждый дескриптор содержит запись в поддерживаемой внутренней таблице. Эти записи содержат адреса ресурсов и средства для идентификации типа ресурса.