1. **Что такое POSIX?**

POSIX - это стандарт, описывающий интерфейс между операционной системой и прикладной программой. Фактически, этот стандарт описывает то, как программа должна вызывать системные функции для того, чтобы свободно запускаться под всеми операционными системами, отвечающими данному стандарту. Сама аббревиатура POSIX расшифровывается как Portable Operating System Interface for UNIX.

1. **Что такое системный вызов?**

Системный вызов — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.

1. **Что такое аппаратное прерывание, программное прерывание?** Прерывание означает временное прекращение основного процесса вычислений для выполнения некоторых запланированных или незапланированных действий, вызываемых работой аппаратуры или программы.

Аппаратные прерывания возникают как реакция микропроцессора на физический сигнал от некоторого устройства (клавиатура, системные часы, клавиатура, жесткий диск и т.д.), по времени возникновения эти прерывания асинхронны, т.е. происходят в случайные моменты времени.

Программные прерывания обрабатываются так же, как аппаратные прерывания. Однако они могут быть сгенерированы только процессами, которые в данный момент выполняются.

Программное прерывание говорит только с ядром операционной системы. Ядро отвечает за планирование любых других процессов, которые должны быть запущены.

1. **Что такое процесс?**

Процесс OS – единица работы OS - объект ядра OS + адресное пространство.

Процесс — это в выполняемая в данный момент программа.

1. **Что такое контекст процесса?**

Контекст процесса – данные, которые сохраняются при переключении процессов и предназначенные для продолжения работы.

Контекст включает содержимое пользовательского адресного пространства - пользовательский контекст (т.е. содержимое сегментов программного кода, данных, стека, разделяемых сегментов и сегментов файлов, отображаемых в виртуальную память), содержимое аппаратных регистров - регистровый контекст (таких, как регистр счетчика команд, регистр состояния процессора, регистр указателя стека и регистров общего назначения), а также структуры данных ядра (контекст системного уровня), связанные с этим процессом.

1. **Что такое родительский и дочерний процесс?**

Родительский процесс — это процесс или приложение, которое может запускать другие приложения.

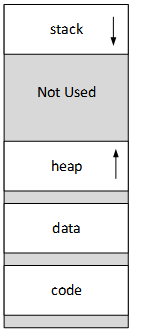
Дочерний процесс – это процесс, созданный другим процессом (родительский процесс).

1. **Что такое процесс инициализации OS?**

**в OS есть процесс инициализации (родитель для всех);**

1. **Перечислите области памяти процесса и поясните их назначение.**

* **Text** или **Code segment**: содержит исполняемые инструкции
* **Heap**: участок динамического выделения памяти
* **Data segment**: различные переменные, в свою очередь делится на две части:
  + **Initialized data segment**: инициализированные данные
  + **Uninitialized data segment**: неициализированные данные — не содержит данных, а только указание на выделение памяти при запуске
* **Stack**: участок памяти, содержащий временные данные



1. **Чем отличаются системные процессы от пользовательских?**

Системные процессы - процессы запускаемые автоматически при запуске OS; Windows: windows-сервисы; Linux-демоны.

Системные процессы являются частью ядра ОС и всегда расположены в оперативной (основной) памяти.

Выполняемые инструкциии данные этих процессов находятся в ядре системы, и поэтому они могут вызывать функции и обращаться к данным, недоступным для остальных процессов, например диспетчер страничного замещения, диспетчер памяти ядра, диспетчер буферного кэша и другие.

Прикладные процессы, как правило, порождаются в рамках пользовательского сеанса. Они могут выполняться как в интерактивном, так и в фоновом режимах.

Процессы, которые выполняют системный код, называются системными и применяются к системе в целом. Они занимаются выполнением таких служебных задач, как распределение памяти, обмен страницами между внутренним и вспомогательным запоминающими устройствами, контроль устройств и т.п. Они также выполняют некоторые задачи «по поручению» пользовательских процессов, например, делают запросы на ввод-вывод данных, выделяют память и т.д. Пользовательские процессы выполняют собственный код и иногда обращаются к системным функциям. Выполняя собственный код, пользовательский процесс пребывает в пользовательском режиме (user mode). В пользовательском режиме процесс не может выполнять определенные привилегированные машинные команды.

1. **Что такое Windows-сервисы, Linux-демоны?**

При запуске OS некоторые процессы (Windows-сервисы, Linux-демоны) загружаются и стартуют автоматически, как правило используются для внутреннего назначения.

Службы Windows (англ. Windows Service, сервисы) — приложения, автоматически запускаемые системой при запуске Windows и выполняющиеся вне зависимости от статуса пользователя. Имеет общие черты с концепцией демонов в Unix.

1. **С помощью каких системных вызовов можно создать дочерний процесс в Windows?**

В Windows вызов всего одной функции CreateProcess управляет и созданием процесса и запуском нужной в ней программы.

После создания нового процесса, родительский и дочерний процессы имеют собственные различные адресные пространства. В тоже время, созданный процесс может использовать одинаковые ресурсы с родительским процессом, например открытые файлы.

1. **С помощью каких системных вызовов можно создать дочерний процесс в Linux? Поясните разницу.**

В UNIX существует только один системный запрос, направленный на создание процесса: fork.

Этот запрос создает дубликат вызываемого процесса. После выполнения запроса fork двум процессам - родительскому и дочернему - соответствуют одинаковые образы памяти, строки окружения и открытые файлы. Обычно, дочерний процесс выполняет системный вызов exec для изменения образа памяти и запуска новой программы.

Функция system() определена в стандартной библиотеке языка С и позволяет вызывать из программы системную команду, как если бы она была набрана в командной строке.

1. **Какие потоки данных доступны любому процессу автоматически?**

стандартные потоки ввода/вывода процесса: потоки имеющие зарезервированные номера - дескрипторы (номера), поток ввода (0), поток вывода (1), поток вывода ошибок (2).

1. **Поясните назначение системного вызова WaitForSingleObject в Windows-приложении.**

WaitForSingleObject(hHandle, dwTimeout);

WaitForSingleObject ждет, когда объект, хэндл которого был передан первым аргументом, перейдет в сигнальное состояние (signaled state).

Если hHandle представляет собой хэндл мьютекса, процедура ждет, когда мьютекс освободится, а затем лочит его. Если же hHandle является хэндлом потока, то процедура просто ждет его завершения. Второй аргумент задает время ожидания в миллисекундах. Можно ждать вечно, передав специальное значение INFINITE. Если указать ноль, процедура не переходит в режим ожидания, а возвращает управление немедленно.

В случае ошибки процедура возвращает WAIT\_FAILED, а подробности поможет узнать GetLastError. В случае успеха возвращается WAIT\_OBJECT\_0, если мы дождались перехода объекта в сигнальное состояние, и WAIT\_TIMEOUT, если отвалились по таймауту. Также мы можем получить WAIT\_ABANDONED. Это происходит в случае, если нить, державшая мьютекс, завершилась, не освободив его. В этом случае мьютекс становится залочен текущей нитью, но целостность данных, доступ к которым ограничивался мьютексом, по понятным причинам находится под вопросом.

1. **Поясните назначение системного вызова wait в Linux-приложении.**

Самая простая функция в семействе называется wait(). Она блокирует вызывающий процесс до тех пор, пока один из его дочерних процессов не завершится (или не произойдет ошибка). Код состояния потомка возвращается через аргумент, являющийся указателем на целое число. В этом коде зашифрована различная информация о потомке. Например, макрос WEXITSTATUS() возвращает код завершения дочернего процесса. Макрос WIFEXITED() позволяет узнать, как именно завершился процесс: обычным образом (с помощью функции exit() или оператора return функции main()) либо аварийно вследствие получения сигнала. В последнем случае макрос WTERMSIG() извлекает из кода завершения номер сигнала.

1. **Дайте развернутое определение процесса OS.**

Процесс OS – единица работы OS - объект ядра OS + адресное пространство.

Процессом, по существу, является программа во время ее выполнения.

1. **OS:** основные свойства процесса:

* процессу соответствует исполняемый программный файл;
* у процесса есть PID;
* у процесса есть Parent PID;
* в Windows: HANDLE – идентификатор объекта OS;
* в OS есть процесс инициализации (родитель для всех);
* запуск и управление (создать, остановить,…) процессом осуществляется с помощью системных вызовов;
* процессы изолированы друг от друга;
* процессу выделяется линейное адресное пространство (размер зависит от разрядности), сегменты: code, static, data, heap, stack;
* контекст процесса – данные, которые сохраняются при переключении процессов и предназначенные для продолжения работы;
* процессу автоматически доступны три потока: ввода, вывода, вывод ошибок.
* при запуске OS некоторые процессы (Windows-сервисы, Linux-демоны) загружаются и стартуют автоматически, как правило используются для внутреннего назначения;
* в составе ОS есть таблица, содержащая объекты ядра процессов (состояние, приоритет, указатели на другие объекты); есть средства OS позволяющие ее просматривать;

процесс – единица работы OS