# **ОСИ 3 ВОПРОСЫ**

1. **Что такое POSIX?**

Portable Operating System Interface – переносимый интерфейс ОС – набор стандартов взаимодействия между ОС и прикладной программой.

Используется для совместимости UNIX-подобных систем, но может быть использован и на не-UNIX системах.

Благодаря этим стандартам можно запускать код на разных UNIX-подобных системах.

1. **Что такое системный вызов?**

Механизм вызова прикладной программой функция ядра ОС методом системного прерывания. Функции ядра ОС кстати работают в привилегированном режиме супервизора (или режиме гипервизора, если такой возможен). В этом режиме доступна вся память и возможно выполнение всех команд процессора.

На RISC/x86 для этого использовался int, на новых системах:

Intel x86\_64 – SYSENTER/SYSEXIT, AMD – SYSCALL/SYSRET

1. **Что такое аппаратное прерывание, программное прерывание?**

*Вообще, прерывание – это запрос на остановку текущего выполняемого кода, чтобы выполнить определенное действие, описанное в обработчике* *прерывания*

**Аппаратные** – реакция микропроцессора на физический сигнал от некоторого устройства (клавиатура, часы, клавиатура и т.д.).

Являются асинхронными, т.е. происходят в случайные моменты времени.

**Программные** – вызываются искусственно с помощью команды из соответствующей программы (например, int)

Предназначены для выполнения некоторых действий операционной системы.

Являются синхронными.

1. **Что такое процесс?**

Единица работы ОС. Объект ядра ОС + адресное пространство.

1. **Что такое контекст процесса?**

В контексте сохраняется вся информация, необходимая для продолжения процесса после его приостановки.

Включает в себя:

* пользовательский контекст (сегменты программного кода, данных, стека, содержимое виртуального адресного пространства)
* регистровый контекст
* контекст системного уровня: *статическая* (PID, PPID, состояние, приоритет) и *динамическая часть* (один или несколько стеков для выполнения в режиме пользователя и режиме супервизора)

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

**Родительский** – тот, который создает новый процесс (дочерний)

**Дочерний** – тот, который создается из некоторого уже работающего процесса. В качестве PPID присваивается PID процесса, который создал дочерний процесс.

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

Процесс, запускаемый при старте ОС; является родительским для всех остальных.

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

*Адреса возрастают от CODE до STACK.*

CODE – инструкции ЦП скомпилированной программы, создаются автоматически. Read-only память

STATIC – сохраняется на протяжении всей жизни программы, используется, например, для глобальных переменных. Часто имеет размер 4 байта. Хранится в отделах .bss и .data

HEAP – тут хранятся основные данные. Динамическая обширная память

DATA – содержит глобальные и статические переменные

STACK – контекст функции: аргументы, возвращаемое значение, адрес возврата. Тут хранятся локальные переменные. Эта память автоматически выделяется ЦП. Часто имеет размер 8 MB.

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

Системные – запускаемые системой (демоны/сервисы), пользовательские – пользователем.

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

Это процессы, которые загружаются и стартуют автоматически, при запуске системы. Пользователь не должен ничего делать для их запуска и работы; все происходит автоматически.

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

CreateProcess() – (было бы неплохо знать, что это просто макрос, который скрывает две функции: CreateProcessA() и CreateProcessW(), которые отличаются только кодировкой – ANSI и Unicode соответственно).

CreateProcessAsUser() – дочерний процесс запускается в контексте системы безопасности пользователя, обозначенного параметром hToken.

CreateProcessWithTokenW() – дочерний процесс с привилегией NT Authority/SYSTEM. Может загружать профили пользователей и работать с WindowsStation и рабочим столом Winsta0 вместо вас (че бля? но я больше ниче не нашел мне впадлу)

CreateProcessWithLogonW() – дочернему процессу не нужно вызывать функцию LogonUser, чтобы подтвердить подлинность пользователя и получить маркер безопасности.

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

**fork()** – создает максимально подробную копию родительского процесса (копирует дескрипторы и т.д.). Создает дочерний процесс. Дочерний процесс получает новый Pid (обычно PPID+1). Возвращает -1 (ошибка), 0 (дочерний процесс), PID дочернего (родительский процесс).

**system()** – в отличие от exec(), Он разветвляется и запускает дочерний процесс, пока родительский ожидает. Под капотом это fork()+exec()+wait()

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

Стандартные потоки процесса — это потоки процесса, имеющие зарезервированный номер (дескриптор) для выполнения некоторых стандартных функций:

0 – поток ввода stdin

1 – поток вывода stdout

2 – поток вывода ошибок stderr

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

Функция библиотеки Window.h. Принимает в качестве параметров HANDLE – дескриптор объекта и DWORD – количество миллисекунд. Функция показывает, что родительский процесс (тот, в котором эта функция была вызвана) будет ожидать сигнального состояния параметра HANDLE (дочернего процесса) до истечения интервала ожидания n миллисекунд.

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

Родительский процесс *ожидает* изменения состояния в дочернем процессе (его приостановки, возобновления, прекращения). Wait(0) [или wait(NULL)] будет ожидать завершения дочернего процесса.

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

Совокупность взаимосвязанных системных ресурсов, расположенных в адресном пространстве, в контексте которой организуется выполнение потоков. Если компьютерная программа сама по себе — лишь пассивная последовательность инструкций, то процесс — непосредственное выполнение этих инструкций.

**Свойства процессов:**

* Создается ядром по системному вызову
* Может создавать дочерние процессы (с помощью системного вызова)
* При создании выделяется адресное пространство
* Соответствует исполняемому файлу (exe)
* Процессы изолированы друг от друга
* При приостановке или переключении процесса вся необходимая для продолжения информация сохраняется в контексте
* Имеет PID и PPID
* Для обмена данными между процессами используется IPC – механизм межпроцессного взаимодействия
* Процессу автоматически отводится три потока (0, 1, 2)
* При запуске OS некоторые процессы загружаются и стартуют автоматически (Windows – сервисы, UNIX –демоны)
* Есть процесс инициализации – родительский для всех процессов