**Что такое фреймворк OS?**

Набор библиотек OS + API интерфейс.

Нужен для взаимодействия OS core и Frameworks.

**Что такое POSIX? Набор стандартов**

POSIX (англ. Portable Operating System Interface — переносимый интерфейс операционных систем) — 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.

**Что такое аппаратное 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.

**Что такое программное прерывание?**

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.

**Что такое системный вызов?**

Систе́мный вы́зов (англ. system call) в программировании и вычислительной технике — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции. С точки зрения программиста, системный вызов обычно выглядит как вызов подпрограммы или функции из системной библиотеки. Однако системный вызов, как частный случай вызова такой функции или подпрограммы, следует отличать от более общего обращения к системной библиотеке, поскольку последнее может и не требовать выполнения привилегированных операций

**Что такое процесс OS?**

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

Проце́сс — это в выполняемая в данный момент программа.

**Что такое контекст процесса OS?**

Каждому процессу соответствует контекст, в котором он выполняется. Этот 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.

**Что такое адресное пространство процесса?**

для всех процессов выполняется виртуализация ресурсов памяти, то есть для каждого процесса создается иллюзия того, что он один использует всю физическую память в системе. Адресное пространство процесса состоит из диапазона адресов, которые выделены процессу, и, что более важно, в этом диапазоне выделяются адреса, которые процесс может так или иначе использовать. Размер адресного пространства зависит от аппаратной платформы. Обычно для каждого процесса существует свое адресное пространство. Адрес памяти в адресном пространстве одного процесса не имеет никакого отношения к такому же адресу памяти в адресном пространстве другого процесса. Тем не менее несколько процессов могут совместно использовать одно общее адресное пространство. Такие процессы называются потоками.

**Перечислите области памяти адресного пространства процесса и поясните их назначение.**

Стандартное распределение памяти процесса выглядит следующим образом, от меньшего адреса к более высокому (или «снизу вверх»): code, static, data, heap, stack;

(на рисунке порядок неверный - heap и data перепутаны местами)



* **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

**Перечислите системные вызовы Windows для создания процесса?**

Функция CreateProcess() (или NtCreateProcess() )создает новый процесс, который выполняется независимо от процесса создания. Если функция CreateProcess завершается без ошибки, то возвращается структура (struct), содержащая дескрипторы и идентификаторы для нового процесса и его основного потока. Процесс порождается непосредственно по желанию, он не должен быть обязательной копией текущего, поэтому не соблюдается родство.

**Перечислите системные вызовы Linux для создания процесса? Только Fork().**

Процессы создаются через две функции Fork() и exec(). Начинается с Fork(), он создает точный клон вызывающего процесса, так называемый «дочерний» процесс. Менеджер исполнения exec() заменяет образ процесса этого клона новой программой, которая должна быть выполнена. Так сложилось исторически и другого способа породить процесс нет, поэтому существует иерархия, как основа основ. После создания у родительского и дочернего процессов возникают собственные разные адресные пространства.

**С помощью каких утилит можно увидеть перечень процессов в Windows?**

tasklist.exe (консольная)

taskmgr.exe (gui)

**С помощью каких утилит можно увидеть перечень процессов в Linux?**

ps, top

**Перечислите свойства процесса OS.**

* 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.

Все процессы могут характеризоваться по различным признакам.

Классификация по времени существования:

1. Реального времени. Жестко исполняются в течение определенного заданного времени.

2. Интерактивные процессы

3. Все остальные

Классификация процессов по происхождению:

1. Порождающие. Задает некоторые требования или условия.

2. Порожденные. Создается по данным требованиям или условиям.

Классификация процессов по динамическому признаку:

1. Последовательные. Интервалы времени существования процессов не пересекаются.

2. Параллельные. Строгое совпадение по времени моментов начала и завершения процессов.

3. Комбинированные. Т1 и Т2 не совпадают, но есть частичное перекрытие.

Классификация процессов по принадлежности к ЦП:

1. Внутренние. Процесс развивается на уровне ЦП

2. Внешние. Развитие под контролем ОС.

Классификация по принадлежности к ОС:

1. Системные

2. Прикладные

Классификация по связности:

1. Изолированные. Не поддерживают никакие варианты связей.

2. Информационно независимые. Используют совместно некоторые ресурсы, но информационно между собой не связаны. Между ними функциональная или временная связи.

3. Взаимодействующие. Процессы с информационными связями. Информационные связи могут быть по-разному реализованы: передача параметров, обмен сообщениями, общие структуры данных

4. Конкурирующие. Характеризуются взаимоисключающими требованиями к совместно используемым ресурсам.