Операционная система (ОС):

Операционная система - это программное обеспечение, которое управляет аппаратными ресурсами компьютера и предоставляет интерфейс для взаимодействия пользователя с компьютером. Она выполняет ряд ключевых функций, включая управление файлами, планирование задач, обеспечение безопасности и управление памятью. Операционная система служит посредником между пользователем и аппаратными устройствами компьютера, обеспечивая удобное и эффективное использование компьютера.

Драйвер:

Драйвер - это программное обеспечение, которое обеспечивает взаимодействие операционной системы с конкретным аппаратным устройством или дополнительным оборудованием. Драйверы позволяют операционной системе распознавать, управлять и взаимодействовать с аппаратными компонентами, такими как принтеры, видеокарты, звуковые карты и другие устройства. Они переводят команды и запросы операционной системы в инструкции, понимаемые конкретным оборудованием.

BIOS (Basic Input/Output System):

BIOS - это набор низкоуровневых программ, хранящихся на чипе материнской платы компьютера. BIOS выполняет ряд важных функций, таких как инициализация и самодиагностика аппаратных устройств при включении компьютера, определение порядка загрузки операционной системы с устройств хранения данных (например, жесткого диска или USB-накопителя) и предоставление интерфейса для настройки некоторых параметров системы. BIOS также может быть обновлен для поддержки нового оборудования или исправления ошибок.

Виды операционных систем:

Существует несколько видов операционных систем, включая:

a. Операционные системы для персональных компьютеров (PC OS): Примеры включают в себя Microsoft Windows, macOS и различные дистрибуции Linux.

b. Операционные системы для серверов: Они оптимизированы для управления серверными ресурсами и обеспечения надежной работы серверных приложений. Примеры включают в себя Windows Server, Linux (например, CentOS, Ubuntu Server) и другие.

c. Мобильные операционные системы: Они разработаны для мобильных устройств, таких как смартфоны и планшеты. Примеры включают в себя Android, iOS и HarmonyOS.

d. Встроенные операционные системы: Они используются во встроенных системах управления, таких как микроконтроллеры и устройства Интернета вещей (IoT). Примеры включают в себя FreeRTOS, Embedded Linux и другие.

e. Реального времени (Real-time) операционные системы: Они предназначены для систем, где требуется точное и предсказуемое время реакции на события в реальном времени. Примеры включают в себя QNX, VxWorks и RTLinux.

Каждый из этих типов операционных систем разработан для удовлетворения определенных потребностей и сценариев использования.

Лаб2

**Задание 06.ответьте на следующие вопросы**

1. Что такое фреймворк OS?
2. Что такое POSIX?
3. Что такое аппаратное прерывание?
4. Что такое программное прерывание?
5. Что такое системный вызов?
6. Что такое процесс OS?
7. Что такое контекст процесса OS?
8. Что такое адресное пространство процесса?
9. Перечислите области памяти адресного пространства процесса и поясните их назначение.
10. Что такое стандартные потоки процесса?
11. Перечислите системные вызовы Windows для создания процесса?
12. Перечислите системные вызовы Linux для создания процесса?
13. С помощью каких утилит можно увидеть перечень процессов в Windows?
14. С помощью каких утилит можно увидеть перечень процессов в Linux?
15. Перечислите свойства процесса OS.
    * + 1. Фреймворк OS (Operating System Framework) - это набор программных компонентов, библиотек и инструментов, предназначенных для разработки операционных систем или создания приложений, работающих на операционных системах.
        2. POSIX (Portable Operating System Interface) - это набор стандартов и спецификаций, разработанных для обеспечения совместимости между различными операционными системами и обеспечения переносимости приложений между ними. POSIX определяет API (Application Programming Interface) и интерфейсы командной строки для унификации функциональности операционных систем, таких как UNIX и UNIX-подобные системы.
        3. Аппаратное прерывание (Hardware Interrupt) - это механизм, который генерирует сигнал аппаратурой компьютера (например, процессором или устройствами ввода/вывода), используются для обработки событий, требующих мгновенного вмешательства, таких как обработка нажатия клавиши на клавиатуре или завершение операции чтения данных с диска.
        4. Программное прерывание (Software Interrupt) - это механизм, позволяющий искусственно вызвать прерывание в программном коде, обычно с целью вызвать выполнение определенного обработчика прерывания.
        5. Системный вызов (System Call) - обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.
        6. Процесс OS (Operating System Process) - это экземпляр выполняемой программы в операционной системе. Каждый процесс имеет свою собственную память, стек вызовов и ресурсы, такие как файлы и сетевые соединения.

Процесс в операционной системе (OS) представляет собой исполняющуюся программу с уникальным идентификатором (PID). Каждый процесс имеет свой код, данные, стек, регистры и другие ресурсы. Операционная система управляет процессами, обеспечивает их изоляцию и взаимодействие через механизмы IPC. Процессы создают дерево, а каждый имеет уникальный PID для идентификации в системе.

* + - 1. Контекст процесса OS (Process Context) - это состояние процесса, включая значения регистров, содержимое памяти и другие ресурсы, необходимые для его выполнения. Контекст процесса позволяет операционной системе переключать выполнение между разными процессами.

Контекст процесса позволяет операционной системе сохранять и восстанавливать состояние процесса при его переключении, а также управлять ресурсами и выполнением процессов в системе.

Контекст процесса в операционной системе - это совокупность данных, определяющих текущее состояние выполнения конкретного процесса, включая значения регистров процессора, состояние памяти, стек вызовов и другие параметры, необходимые для управления выполнением процесса.

* + - 1. Адресное пространство процесса (Process Address Space) - это диапазон адресов памяти, который выделен для определенного процесса. Оно содержит код процесса, данные, стек вызовов и другие ресурсы, необходимые для выполнения приложения.
      2. Области памяти адресного пространства процесса могут включать:

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.

Сегмент файлового дескриптора (File Descriptor Segment): Содержит информацию о файлах, открытых процессом.

* + - 1. Стандартные потоки процесса - это три потока данных, связанных с каждым процессом в операционной системе:

stdin (стандартный ввод): Поток для ввода данных в процесс.

stdout (стандартный вывод): Поток для вывода данных из процесса.

stderr (стандартный вывод ошибок): Поток для вывода сообщений об ошибках из процесса.

Системные вызовы Windows для создания процесса включают:

CreateProcess: Создание нового процесса.

CreateThread: Создание нового потока внутри процесса.

Системные вызовы Linux для создания процесса включают:

fork: Создание копии текущего процесса.

exec: Замена текущего процесса новым процессом.

* + - 1. В Windows утилитой для просмотра списка процессов является "Task Manager" (Диспетчер задач).

В Linux утилитой для просмотра списка процессов является "ps" (Process Status) или "top" (динамический мониторинг процессов).

Свойства процесса OS могут включать в себя:

Идентификатор процесса (PID).

Статус процесса (запущен, завершен и т. д.).

Используемые системные ресурсы (память, процессорное время).

Родительский процесс (процесс, создавший данный процесс).

Приоритет выполнения и приоритет ввода-вывода.

Рабочий каталог процесса и другие атрибуты.

Bat-файл – это пакетный файл Windows

Пакетный файл — текстовый файл Windows, содержащий последовательность команд, предназначенных для исполнения командным интерпретатором.

Ядро операционной системы — это самая низкоуровневая часть программного обеспечения, которая представляет собой основу ОС и является связующим звеном между аппаратным и программным обеспечением компьютера.

Ядро́ (англ. kernel) — центральная часть операционной системы (ОС), обеспечивающая приложениям координированный доступ к ресурсам компьютера, таким как процессорное время, память, внешнее аппаратное обеспечение, внешнее устройство ввода и вывода информации.

Лаб3

**1.Что такое POSIX?**

POSIX (Portable Operating System Interface) - это набор стандартов и спецификаций, разработанных для обеспечения совместимости между различными операционными системами и обеспечения переносимости приложений между ними. POSIX определяет API (Application Programming Interface) и интерфейсы командной строки для унификации функциональности операционных систем, таких как UNIX и UNIX-подобные системы.

1. **Что такое системный вызов?**

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

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

Аппаратное прерывание (Hardware Interrupt) - это механизм, который генерирует сигнал аппаратурой компьютера (например, процессором или устройствами ввода/вывода), используются для обработки событий, требующих мгновенного вмешательства, таких как обработка нажатия клавиши на клавиатуре или завершение операции чтения данных с диска.

Программное прерывание (Software Interrupt) - это механизм, позволяющий искусственно вызвать прерывание в программном коде, обычно с целью вызвать выполнение определенного обработчика прерывания.

1. **Что такое процесс?**

Процесс OS (Operating System Process) - это экземпляр выполняемой программы в операционной системе. Каждый процесс имеет свою собственную память, стек вызовов и ресурсы, такие как файлы и сетевые соединения.

1. **Что такое контекст процесса?**

Контекст процесса OS (Process Context) - это состояние процесса, включая значения регистров, содержимое памяти и другие ресурсы, необходимые для его выполнения. Контекст процесса позволяет операционной системе переключать выполнение между разными процессами.

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

Родительский процесс - это процесс, который порождает другие процессы. Дочерний процесс - это процесс, порожденный другим процессом. Родительский процесс может продолжать работу, в то время как его дочерний процесс выполняется. Пример взаимодействия - использование **fork()** в языке C для создания нового процесса.

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

Процесс инициализации операционной системы (OS) представляет собой первый процесс, который запускается при загрузке компьютера и инициирует запуск всех остальных процессов и служб операционной системы.

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

Области памяти адресного пространства процесса могут включать:

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.

Сегмент файлового дескриптора (File Descriptor Segment): Содержит информацию о файлах, открытых процессом.

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

Системные процессы обладают высокими привилегиями, управляют операционной системой и скрыты от пользователя. Пользовательские процессы выполняют задачи конечного пользователя, ограничены в привилегиях, видны пользователю и взаимодействуют с ним через интерфейс приложений.

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

Windows-сервисы и Linux-демоны - это фоновые процессы в операционных системах Windows и Linux соответственно. Они предназначены для выполнения задач в фоновом режиме, таких как обеспечение системных служб или служебных функций. Windows-сервисы работают в среде Windows и управляются специальными инструментами, в то время как Linux-демоны выполняются в среде Linux и управляются системами инициализации, такими как systemd или init.d.

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

Два основных системных вызова для создания дочернего процесса в Windows - это **CreateProcess** и **CreateProcessAsUser**

**CreateProcess:** Создает новый процесс и его первичный поток.

* **Использование:** Основной вызов для создания дочернего процесса в Windows.

**CreateProcessAsUser:** Аналог **CreateProcess**, который позволяет создать процесс от имени другого пользователя.

* **Использование:** Часто используется в контексте служб Windows.

CreateProcess: Создает процесс от имени текущего пользователя, который выполняет вызов.

CreateProcessAsUser: Позволяет создавать процесс от имени другого пользователя, предоставляя соответствующий токен доступа.

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

В Linux для создания дочернего процесса можно использовать системный вызов **fork()** в сочетании с **exec()** или их комбинацией.

**fork():** Создает точную копию вызывающего процесса, которая становится дочерним процессом.

* **Использование:** **fork()** возвращает 0 в дочернем процессе и идентификатор дочернего процесса в родительском процессе.

**exec():** Заменяет текущий процесс новым процессом, загружая новый исполняемый файл.

* **Использование:** Существует несколько функций **exec**, таких как **execl**, **execv**, **execle** и другие, которые различаются по способу передачи аргументов и переменной окружения.

**Разница между fork() и exec():**

* **fork():** Создает дочерний процесс как точную копию родительского процесса. Все переменные и состояние родительского процесса копируются в дочерний процесс.
* **exec():** Заменяет адресное пространство текущего процесса новым исполняемым файлом. По сути, происходит полная замена кода, данных и стека процесса.

Обычно **fork()** используется для создания дочернего процесса, а затем **exec()** используется внутри дочернего процесса для загрузки нового исполняемого файла и замены его текущего содержимого. Это позволяет создавать дочерние процессы с различной функциональностью относительно родительского процесса.

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

Любой процесс автоматически имеет доступ к трем стандартным потокам данных:

1. **stdin (стандартный ввод):** Предназначен для ввода данных в программу (идентификатор 0).
2. **stdout (стандартный вывод):** Предназначен для вывода данных из программы (идентификатор 1).
3. **stderr (стандартный поток ошибок):** Предназначен для вывода сообщений об ошибках (идентификатор 2).
4. **Поясните назначение системного вызова WaitForSingleObject в Windows-приложении.**

**WaitForSingleObject** в Windows используется для синхронизации потоков/процессов. Он блокирует текущий поток до появления сигнала от объекта синхронизации (например, мьютекса или события). Может использоваться для ожидания завершения другого потока, получения доступа к ресурсу и других сценариев синхронизации.

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

wait в Linux используется родительским процессом для блокировки и ожидания завершения дочернего процесса. Возвращает идентификатор завершенного процесса и статус его завершения. Родительский процесс может использовать эту информацию для обработки завершения дочернего процесса.

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

Процесс OS (Operating System Process) - это экземпляр выполняемой программы в операционной системе. Каждый процесс имеет свою собственную память, стек вызовов и ресурсы, такие как файлы и сетевые соединения.