### 1. Уровни привилегий, режим пользователя и режим ядра:

#### Уровни привилегий:

В мире Linux, уровни привилегий классифицируются с использованием числовых значений, известных как идентификаторы уровней привилегий (Privilege Level Identifiers). Обычно используются следующие уровни:

- Уровень 0 (ноль) - Ядро (Kernel):

- Этот уровень предоставляет полный доступ ко всем ресурсам системы и является наивысшим уровнем привилегий. Только ядро операционной системы работает на этом уровне.

- Уровни 1-2:

- Зарезервированы для использования ядра и обычно не применяются.

- Уровень 3 - Режим пользователя с привилегиями (User Mode with Privileges):

- Обычный режим пользователя, но с некоторыми дополнительными привилегиями. На этом уровне работают пользовательские программы.

- Уровни 4-15:

- Зарезервированы для будущего использования.

#### Режим пользователя и режим ядра:

- Режим пользователя (User Mode):

- В этом режиме работают обычные пользовательские приложения, которые ограничены в своем доступе к ресурсам системы. Программы выполняются на уровне привилегий 3.

- Режим ядра (Kernel Mode):

- Ядро операционной системы работает на уровне привилегий 0 и имеет полный доступ ко всем ресурсам системы. В режим ядра переключается при выполнении системных вызовов и обработке прерываний.

#### Переключение между режимами:

Переключение между режимами происходит при выполнении определенных событий, таких как системные вызовы, обработка прерываний и исключений. Программы пользователей выполняются в пользовательском режиме, и только при необходимости (например, для выполнения привилегированных операций) происходит переключение в режим ядра.

2. Системный вызов и реализация:

Системный вызов - это способ взаимодействия пользовательского пространства с ядром операционной системы. Он предоставляет интерфейс для выполнения операций, таких как чтение/запись файлов, создание процессов, управление памятью и другие основные операции. Когда приложение хочет выполнить определенное действие, оно делает системный вызов.

### Реализация системного вызова в Unix/Linux:

1. Вызов:

- Пользовательское приложение делает системный вызов, указывая номер вызова и передавая необходимые параметры.

2. Прерывание или инструкция:

- Происходит переключение в режим ядра с помощью прерывания или специальной инструкции (например, `int 0x80`).

3. Сохранение контекста:

- Контекст текущего процесса (регистры, флаги) сохраняется, чтобы восстановить его позднее.

4. Выполнение системного вызова:

- Ядро выполняет запрошенное действие, используя переданные параметры. Например, чтение данных из файла или создание нового процесса.

5. Возврат результата:

- Результат выполнения системного вызова возвращается пользовательскому пространству (например, через регистр `eax`).

6. Восстановление контекста и возврат управления:

- Контекст процесса восстанавливается, и управление возвращается обратно в пользовательское приложение.

### 3. Монолитная и микроядерная архитектура ОС:

#### Монолитная архитектура:

- Определение:

- В монолитной архитектуре все компоненты операционной системы находятся в едином исполняемом образе, называемом ядром. Ядро выполняет все функции, такие как управление процессами, памятью, файловыми системами и драйверами устройств.

- Преимущества:

- Простота в разработке и понимании.

- Быстрый доступ к функциям ядра, так как они находятся в одном адресном пространстве.

- Недостатки:

- Масштабируемость может быть проблемой при добавлении новых функций.

- Ошибки в ядре могут привести к сбою всей системы.

#### Микроядерная архитектура:

- Определение:

- Микроядерная архитектура разделяет функции операционной системы на минимальные, независимые компоненты, выполняющиеся в привилегированном режиме ядра. Основные функции, такие как управление процессами и памятью, остаются в ядре, а дополнительные службы (файловые системы, драйверы) работают как отдельные процессы в пространстве пользователя.

- Преимущества:

- Легче добавлять и изменять функции системы без изменения всего ядра.

- Повышенная надежность, так как сбои в отдельных компонентах не приводят к сбою всей системы.

- Недостатки:

- Некоторые операции могут быть медленнее из-за переключения контекста между компонентами.

- Большее потребление ресурсов из-за дополнительных коммуникаций между компонентами.

### 4. Структура ОС Linux и ядра:

#### Структура ОС Linux:

1. Ядро Linux:

- Ядро является главной частью операционной системы. Оно обеспечивает базовые службы, такие как управление процессами, управление памятью, управление файловой системой, сетевые функции и драйверы устройств.

2. Пользовательское пространство:

- Включает пользовательские программы и библиотеки. Программы в этом пространстве взаимодействуют с ядром через системные вызовы и библиотеки.

3. Библиотеки:

- Содержат набор функций, облегчающих разработку приложений. Примеры: библиотека стандартных функций (libc), библиотека для работы с сетью (libnet), и другие.

4. Командный интерфейс (Shell):

- Пользовательский интерфейс, который позволяет вводить команды и управлять системой. Примеры: Bash, Zsh.

5. Утилиты и сервисы:

- Различные утилиты и службы, предоставляющие дополнительные возможности. Примеры: системные утилиты (ls, ps), службы (systemd).

#### Структура ядра Linux:

1. Ядро (Kernel):

- Включает подсистемы для управления процессами, файловыми системами, сетевыми протоколами, управления памятью, устройствами и другие.

2. Модули ядра (Kernel Modules):

- Дополнительные компоненты, которые можно загружать и выгружать из ядра во время работы системы. Это позволяет динамически расширять функциональность ядра.

3. Драйверы устройств:

- Обеспечивают взаимодействие между ядром и аппаратным обеспечением. Каждый драйвер предоставляет интерфейс для работы с конкретным типом устройства.

4. Подсистемы:

- Логически сгруппированные части ядра, управляющие определенными аспектами системы. Примеры: планировщик задач, управление памятью, файловая система.

5. Планировщик задач:

- Отвечает за распределение процессорного времени между запущенными процессами.

6. Управление памятью:

- Отвечает за управление физической и виртуальной памятью системы.

7. Файловая система:

- Обеспечивает организацию данных на диске и предоставляет интерфейс для работы с файлами и каталогами.

8. Сетевые протоколы:

- Поддерживают коммуникацию по сети, включая TCP/IP протоколы.

### 5. Типы пользователей:

#### Основные типы пользователей в Linux:

1. Суперпользователь (root):

- Описание: Суперпользователь обладает полными привилегиями и имеет доступ ко всем ресурсам системы. Ему разрешено выполнять любые операции и изменять настройки системы.

- Идентификатор пользователя (UID): 0.

2. Обычные пользователи:

- Описание: Обычные пользователи - это обычные учетные записи, предназначенные для выполнения обычных задач. Они имеют ограниченные привилегии и доступ к определенным ресурсам.

- Идентификаторы пользователя (UID): Различные от 0.

3. Системные пользователи:

- Описание: Это учетные записи, созданные для запуска служб и процессов, которые не связаны с конкретными пользователями. Системные пользователи обычно используются для обеспечения безопасности и разделения обязанностей.

- Идентификаторы пользователя (UID): Назначаются в диапазоне от 1 до 999.

4. Группы пользователей:

- Описание: Пользователи объединяются в группы для облегчения управления правами доступа к файлам и ресурсам. Каждый пользователь может принадлежать одной или нескольким группам.

- Идентификаторы групп (GID): Каждая группа имеет свой уникальный идентификатор.

6. Опишите формат структур работы с файлами: DIR, dirent. Stat

**Описание:** DIR представляет структуру каталога в языке программирования С. Он используется для работы с директориями и содержит указатель на текущую запись и дескриптор открытой директории.

Поля структуры stat включает следующие элементы:

***struct stat {***

***dev\_t st\_dev; /\**** логическое устройство, где находится файл ***\*/***

***ino\_t st\_ino; /\**** номер индексного дескриптора ***\*/***

***mode\_t st\_mode; /\**** права доступа к файлу ***\*/***

***nlink\_t st\_nlink; /\**** количество жестких ссылок на файл ***\*/***

***uid\_t st\_uid; /\**** ID пользователя-владельца ***\*/***

***gid\_t st\_gid; /\**** ID группы-владельца ***\*/***

***dev\_t st\_rdev; /\**** тип устройства ***\*/***

***off\_t st\_size; /\**** общий размер в байтах ***\*/***

***unsigned long st\_blksize; /\**** размер блока ввода-вывода ***\*/***

***unsigned long st\_blocks; /\**** число блоков, занимаемых файлом ***\*/***

***time\_t st\_atime; /\**** время последнего доступа ***\*/***

***time\_t st\_mtime; /\**** время последней модификации ***\*/***

***time\_t st\_ctime; /\**** время последнего изменения ***\*/***

***};***

**struct dirent {**

**ino\_t d\_ino; /\*** inode number **\*/**

**off\_t d\_off; /\*** offset to the next dirent **\*/**

**unsigned short d\_reclen; /\*** length of this record **\*/**

**char d\_name[MAXNAMELEN]; /\*** filename **\*/**

**};**

### 7. Принцип организации файловой системы на основе связанного списка блоков (на примере FAT), структура каталога.

Организация файловой системы на основе связанного списка блоков (FAT):

File Allocation Table (FAT) - это метод организации файловой системы, который использует связанный список блоков для хранения файлов. Он широко применяется в файловых системах, таких как FAT16, FAT32 и других.

Принцип работы:

1. Блоки данных: Хранятся на диске и представляют собой небольшие фрагменты информации. Файл разделяется на блоки, и каждый блок хранится на диске.

2. Связанный список: Каждый блок содержит указатель на следующий блок в цепочке, образуя таким образом связанный список. Последний блок в цепочке имеет указатель, указывающий на конец файла.

3. File Allocation Table (FAT): Это центральная структура, хранящая информацию о том, какие блоки данных принадлежат каким файлам. FAT содержит записи для каждого блока данных, указывая на следующий блок в цепочке данных.

Структура каталога:

1. Каталоги: В FAT системах каталоги представлены как специальные файлы, называемые каталогами. Каталог содержит записи для файлов и подкаталогов.

2. Структура записи в каталоге: Каждая запись в каталоге содержит информацию о файле, включая его имя, атрибуты, размер, дату создания и указатель на первый блок данных.

3. Древовидная структура: Каталоги могут быть вложенными, образуя древовидную структуру файловой системы.

### 8. Принцип организации файловой системы на основе i-nodes (на примере ext). Структура каталога. Структура носителя. Формат i-node.

Организация файловой системы на основе i-nodes (ext):

Extended File System (ext) - это файловая система, используемая в операционных системах Linux.

Принцип работы:

1. i-nodes (индексные узлы): Каждый файл и каталог в ext файловой системе имеет свой i-node, который содержит метаданные файла, такие как права доступа, владелец, размер, даты изменения и список блоков данных.

2. Блоки данных: Фактическое содержимое файла хранится в блоках данных. i-node содержит указатели на блоки данных.

3. Структура файловой системы: Файловая система ext организована в виде групп блоков, где каждая группа содержит i-nodes и данные.

Структура каталога:

1. Каталоги: В ext файловой системе каталоги также представлены как специальные файлы, содержащие записи о файлах и подкаталогах.

2. Структура записи в каталоге: Каждая запись содержит имя файла, его i-node и тип файла.

3. Древовидная структура: Каталоги могут быть вложенными, создавая древовидную структуру.

Структура носителя:

1. Суперблок: Информация о файловой системе, такая как размер блока, количество i-nodes, и другие параметры, хранится в суперблоке.

2. Группы блоков: Файловая система разбивается на группы блоков для более эффективного управления ресурсами.

### 9. Монтирование файловой системы:

Монтирование файловой системы - это процесс подключения (смонтирования) файловой системы на определенное место (каталог) в иерархии директорий, что позволяет системе видеть и работать с данными файлами и каталогами. В Linux это делается с помощью команды `mount`.

Процесс монтирования включает следующие шаги:

- Выбор устройства: Определите файловую систему и устройство, которое вы хотите смонтировать, например, раздел на жестком диске или съемное устройство (USB-накопитель).

- Выбор точки монтирования: Укажите каталог, в который будет смонтирована файловая система. Этот каталог должен существовать.

- Монтирование: Используйте команду `mount` для монтирования файловой системы на выбранную точку монтирования.

sudo mount /dev/sdX /mnt/my\_mount\_point

- Работа с файлами: Теперь вы можете работать с файлами и каталогами в смонтированной файловой системе, как если бы она была частью вашей системы.

- Размонтирование: По завершении работы с файловой системой, необходимо выполнить размонтирование с помощью команды `umount`. Например:

sudo umount /mnt/my\_mount\_point

Монтирование файловых систем позволяет эффективно использовать различные носители и хранить данные в удобном формате, делая их доступными для операционной системы.