Лабораторная работа №2
«Работа с файлами в Astra Linux»

1. Теоретическая часть

1.1. История файловых систем

На заре вычислительной техники первые компьютеры (мэйнфреймы) обрабатывали наборы перфокарт, содержащих программы. Именно они использовались в качестве накопителей информации.

Но неудобство перфокарт заключалось в том, что на доступ к нужной информации требовалось много времени — мало того, что нужно было загрузить перфокарты в считыватель, так ещё и процесс считывания был не быстрым. Если в каком-нибудь институте была ЭВМ, то доступ к ней был расписан на месяцы вперед от заказчиков. Если находили ошибку, то работа машины стопорилась на несколько часов, ведь с перфокарты просто так нельзя удалить данные.

Компания IBM разработала первый прототип жёсткого диска, который выглядел монструозно и состоял из одного магнитного диска с механизмом доступа.

Такие диски назывались *disk files*. Слово англ. *file* происходит от лат. *filum* («нить, струна»). Так файлом называли колоду перфокарт, затем само устройство с диском имело такое название. Более того, регистры процессора, в которых хранятся временные данные, объединены в один регистровый файл.

Спустя время была введена концепция файловой системы, когда на одном запоминающем устройстве существует несколько виртуальных «устройств памяти», что и дало слову «файл» современное значение. Имена файлов состояли из двух частей: «основного имени» и «дополнительного имени». Последнее существует и поныне как расширение имени файла.

По мере развития вычислительной техники файлов в системах становилось всё больше. Для удобства работы с ними, их, как и другие данные, стали организовывать в структуры. Вначале это был простой массив, «привязанный» к конкретному носителю информации. В настоящее время наибольшее распространение получила древовидная организация с возможностью монтирования и вставки дополнительных связей (то есть ссылок). Соответственно, имя файла приобрело характер пути к файлу: перечисление узлов дерева файловой системы, которые нужно пройти, чтобы до него добраться.

1.2. Понятие файловой системы

Файл - это именованная область данных на носителе информации, которая представляется пользователю в виде единого непрерывного блока.

Имя файла, также называемое *filename*, представляет собой строку, которая используется для идентификации файла.

Имена файлов в GNU/Linux могут содержать любые символы, кроме косой черты (/), которая зарезервирована для использования в качестве имени корневого каталога и разделителя каталогов, и символа NUL (\0), который используется для обозначения конца текста. Допускаются пробелы, хотя их

лучше избегать, поскольку в некоторых случаях они могут быть несовместимы с устаревшим программным обеспечением.

Обычно в именах файлов используются только буквенно-цифровые символы (в основном строчные буквы), подчеркивания, дефисы и точки. Другие символы, такие как знаки доллара, процента и скобки, имеют особое значение для оболочки и могут мешать работе с ними. Имена файлов никогда не должны начинаться с дефиса.

Имена файлов в каталоге должны быть уникальными. Однако несколько файлов и каталогов с одинаковым именем могут находиться в разных каталогах, потому что у таких файлов будут разные абсолютные пути, и, таким образом, система сможет их различать.

Однако имена файлов предназначены только для удобства пользователей. Существует разница между тем, как файлы отображаются пользователям, и тем, как они фактически хранятся на компьютере. Отдельные файлы обычно хранятся не в виде непрерывных блоков данных, а скорее в виде множества фрагментов, разбросанных в разных местах на жестком диске или даже на нескольких жестких дисках.

То есть то, каким образом располагаются данные на носителях информации, как именно они считываются и интерпретируются и в каком виде представляются пользователю, определяется файловой системой.

Файловая система - это часть операционной системы, определяющая способ организации, хранения и именования данных на носителях информации в вычислительном устройстве.

Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имён файлов (и каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла.

Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

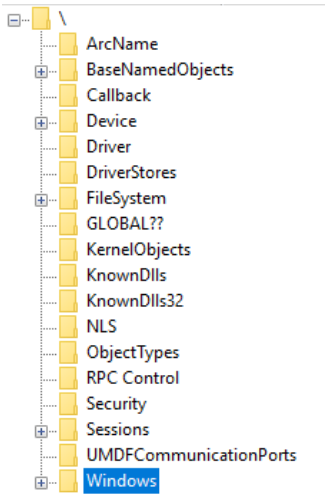
Файловая система связывает носитель информации с одной стороны и API для доступа к файлам — с другой. Когда прикладная программа обращается к файлу, она не имеет никакого представления о том, каким образом расположена информация в конкретном файле, так же как и о том, на каком физическом типе носителя (CD, жёстком диске, магнитной ленте, блоке флеш-памяти или другом) он записан. Всё, что знает программа — это имя файла, его размер и атрибуты.

Однако файловая система необязательно напрямую связана с физическим носителем информации. Существуют виртуальные файловые системы, а также сетевые файловые системы, которые являются лишь способом доступа к файлам, находящимся на удалённом компьютере.

1.3. Файловая система как иерархия каталогов

С точки зрения пользователя, файловая система — это вся иерархия каталогов, также называемая деревом каталогов.

В Windows пользователю не доступен корень файловой системы. То есть абсолютный путь к файлу начинается с имени диска (C, D, E и т.д.). Однако с помощью сторонней программы (например, WinObj) показывается вся структура файловой системы, которая начинается с корня \.



И по факту, путь до файла имеет вид **\Device\HarddiskVolume\Catalog\File.txt**, который для пользователя преобразовывается в **C:\Catalog\File.txt**.

В GNU/Linux нет привычных для пользователя Windows дисков, например диск C:\ или диск D:\. Вместо этого в Linux

существует единое дерево каталогов. Корневой каталог называется «root» и обозначается так (/).

Дистрибутивы GNU/Linux стараются придерживаться стандарта *FHS* (*Filesystem Hierarchy Standard*). Этот стандарт описывает, как должна выглядеть структура каталогов, то есть по какому пути система будет искать определенные файлы. Например, программы будут лежать в одном каталоге, а библиотеки для этих программ в другом каталоге. И программа будет знать в каком каталоге искать библиотеку для своей работы, если программист создававший эту программу придерживался этого стандарта.

По этому стандарту структура каталогов выглядит так:

/ — корневой каталог;

/bin — основные утилиты, которые могут использовать как обычные пользователи так и системные администраторы, но которые необходимы когда не смонтированы другие файловые системы;

/boot — файлы нужные для загрузки системы;

/dev — файлы устройств;

/etc — конфигурационные файлы;

/home — домашние каталоги обычных пользователей;

/lib — библиотеки для программ из каталогов /bin и /sbin и модули ядра;

/media — точки монтирования для сменных носителей;

/mnt — точки монтирования для временного монтирования;

/opt — дополнительное программное обеспечение;

/proc — псевдофайловая система, показывает состояние ядра и запущенные процессы в системе;

/root — домашний каталог пользователя root;

/run — псевдофайловая система, показывает данные, относящиеся к запущенным процессам;

/sbin — системные программы, которые может выполнять только суперпользователь;

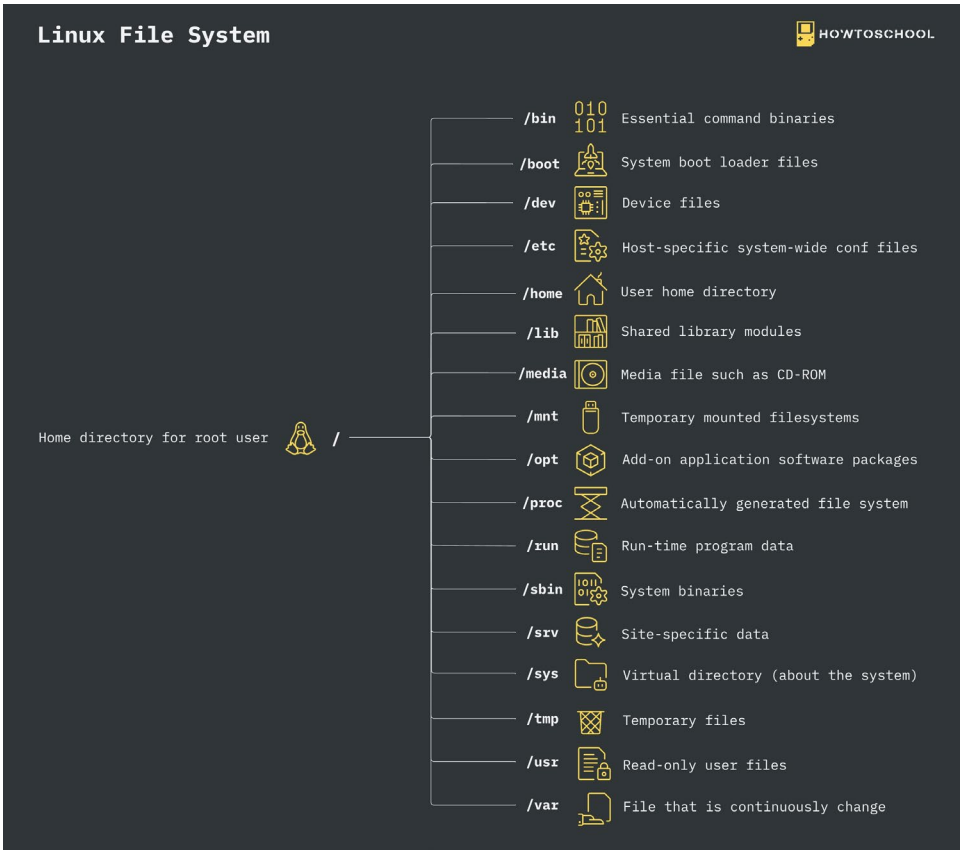
/srv — данные для сервисов предоставляемых системой;

/sys — псевдофайловая система, показывает устройства и драйверы;

/tmp — временные файлы, которые не сохраняются после перезагрузки;

/usr — вторичная иерархия, например /usr/bin и /usr/sbin — дополнительные программы, а /usr/lib — библиотеки для этих программ;

/var — изменяемые файлы (логи, базы данных, почтовые файлы), в этом же каталоге есть /var/tmp — временные файлы, которые должны быть сохранены после перезагрузки.



1.4. Файловая система как способ организации данных

С точки зрения аппаратуры, файловая система — это тип организации хранения данных на накопителе информации.

Каждый тип файловой системы имеет свой собственный набор правил для управления выделением дискового пространства файлам и для связывания данных о каждом файле (называемых **метаданными**) с этим файлом, таких как его имя файла, каталог, в котором он расположен, его разрешения и дата создания. Метаданные хранятся в другом месте файловой структуры, а не в самих файлах.

Современные дистрибутивы GNU/Linux используют файловую систему ***ext4*** (англ. *fourth extended file system, ext4fs*). В ОС Windows используется ***NTFS*** (англ. *new technology file system*). Съёмные накопители (внешние диски, USB FLASH-накопители, SD-карты) хранят файлы по системе ***FAT32*** (от англ. *File Allocation Table — «таблица размещения файлов»*) или ***exFAT*** (при большом объёме памяти).

Файловые системы ***ext*** используют блоки для хранения данных. По умолчанию размер одного блока равен 4096 байта. В начале раздела диска расположен суперблок, в котором находятся метаданные всей файловой системы, а за ним идут несколько зарезервированных блоков, а затем размещена таблица ***Inode*** и только после неё блоки с данными.

В то время как пользователи идентифицируют файлы по их именам, Unix-подобные операционные системы идентифицируют их по индексным дескрипторам (***Inode***).

Inode, I-node или индексный дескриптор — это структура данных, в которой хранятся метаданные файла и перечислены блоки с данными файла.

Они необходимы, потому что у одного файла может быть несколько имен или даже не быть имени. Имя файла в Unix-подобной операционной системе - это просто запись в таблице Inode. Директории - это тоже Inode типа

директория, в которых вместо содержимого файла содержится список имён файлов и номера их Inode.

1.5. Работа с файлами в GNU/Linux

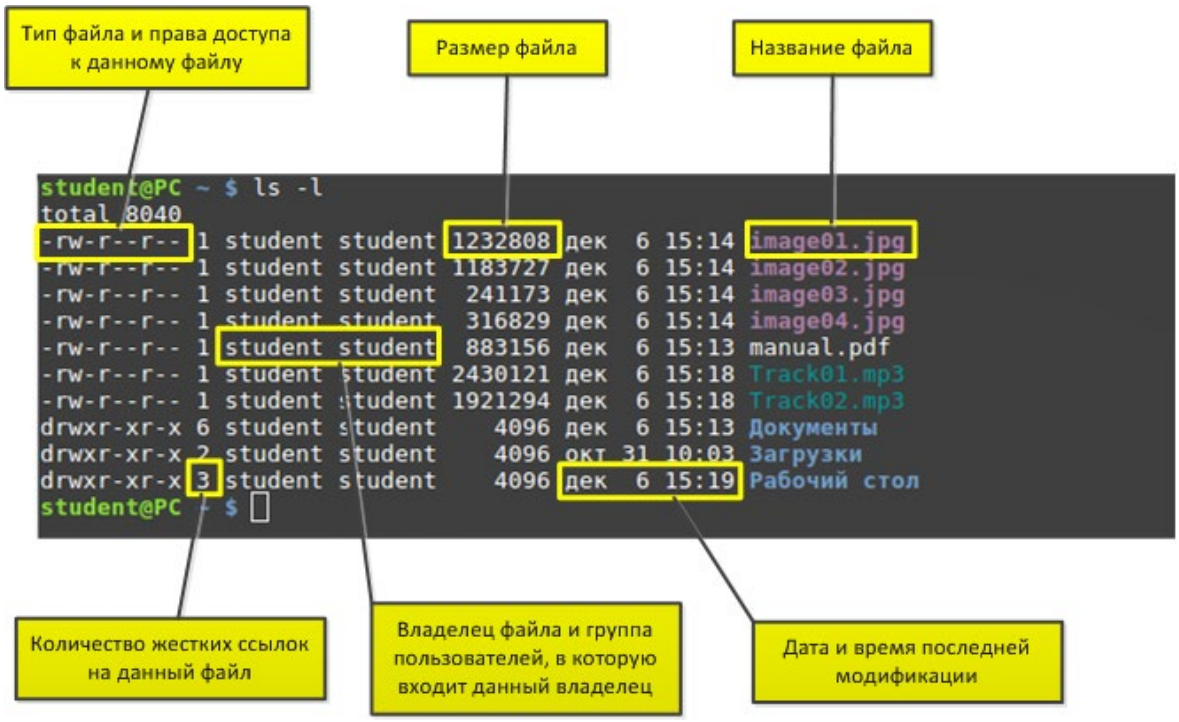
Для работы с файлами в терминале чаще всего используют команды, приведённые в следующей таблице:

№	Команда	Описание	Пример
0	man	Описание работы команды	man ls
1	pwd	Показать текущее местонахождение	~/SAB$ pwd /home/user/SAB
2	ls	Позволяет просмотреть содержимое текущего каталога	~/SAB$ ls 1 1.txt
3	cd <путь к директории>	Перейти в другую директорию	~$ cd ~ /SAB/2 (полный путь) или ~/SAB$ cd 2 (короткий путь)
4	mkdir <название директории>	Создание директории	~/SAB$ mkdir 1
5	touch <название файла>	Создание файла	~/SAB$ touch 1.txt
6	nano <название файла>	Редактирование файла	~/SAB$ nano 1.txt
7	cp <что_копировать> <куда_копировать>	Копирование файла	~/SAB/1$ cp 1.txt ~/SAB/2
8	cp -r <путь_к_папке> <путь_к_новому_месту>	Копирование директории	~/SAB/1$ cp -r 1 ~/SAB/2
9	mv <что_переместить> <куда_переместить>	Переместить файл	~/SAB/1$ mv 1.txt ~/SAB/2

10	rm <название файла>	Удалить файл	~/SAB/1$ rm 1.txt
11	rm -r <название файла>	Удалить директорию	~/SAB/1$ rm -r 1

Для редактирования файла используют различные текстовые редакторы. По умолчанию в дистрибутивы включён редактор GNU nano, также может быть доступен редактор Vim. В nano используются управляющие символы, которые указаны в нижней панели. Для их задействования необходимо зажать клавишу **ctrl** + клавиша, указанная в панели. Например, чтобы выйти из редактора, надо нажать **ctrl + x**.

Чтобы посмотреть, какие файлы доступны в текущей директории (**pwd**), используется команда **ls**. Для вывода подробной информации о файлах используется ключ -l. Будет выведено следующее:



Смысл прав доступа к файлу можно продемонстрировать следующей картинкой:

Чтобы узнать номер Inode файла, используется ключ **-i**. Рассмотрим глубже, каким образом файловая система хранит информацию о файлах и их содержимом.

Создадим в домашней директории папку **Catalog** и перейдём в неё командой

```
mkdir Catalog
cd Catalog
```

В текущей директории создадим файл **file.txt** и запишем в него текст «**Hello, World!**».

Командой **ls -i** узнаем номер **Inode** созданного файла. Например 915727. Введём в командную строку следующее:

```
sudo debugfs /dev/vda1
```

Программа ***debugfs*** (debug file system) используется для отладки работы файловой системы ***ext4***. В качестве файловой системы указан диск, лежащий по пути **/dev/vda1**, в котором хранится система. Чтобы узнать имя диска, воспользуемся командой

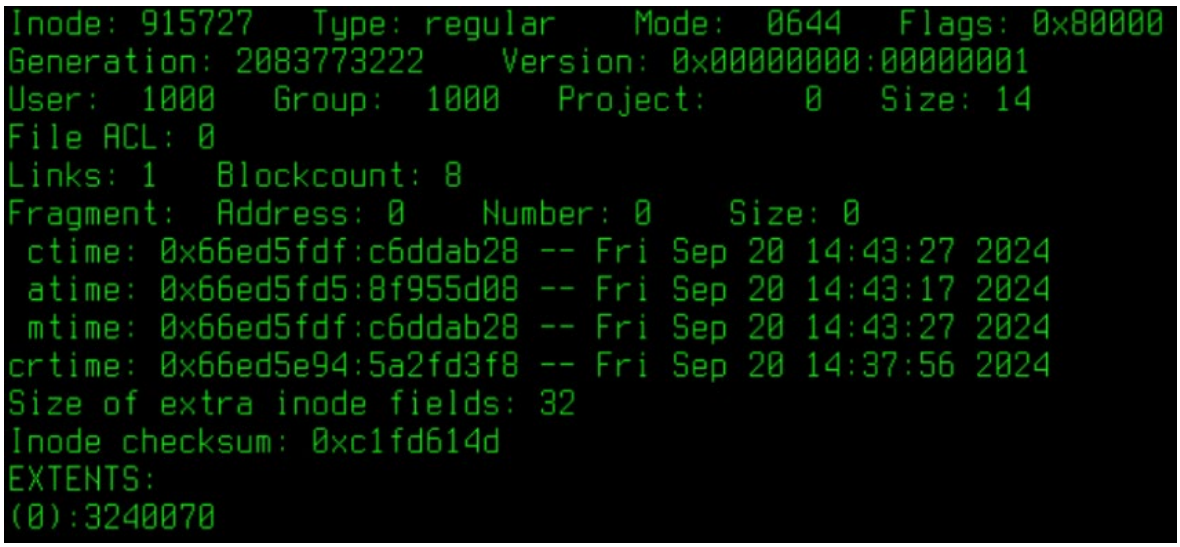
```
df -h
```

Который выведет нам информацию об использовании дискового пространства. Нас интересует диск, который смонтирован в корень \.

Войдя в режим работы программы ***debugfs***, введём команду

```
stat <915727>
```

Эта команда выведет информацию об Inode по указанному в угловых скобках номеру.



Поясним некоторые поля:

- **Inode**: номер инода;
- **Type**: тип файла (regular – обыкновенный файл, могут быть другие типы);
- **Mode**: номер режима прав доступа;
- **User**: номер пользователя, создавшего файл;
- **Group**: номер группы, которой принадлежит файл;
- **ctime**: время последней модификации прав доступа или владельца;
- **atime**: время последнего обращения к файлу;
- **mtime**: время последнего изменения содержимого;
- **Extens**: номера блоков данных файла.

В примере показано, что созданный файл содержит блок данных с номером 3240070. Выйдем из окна клавишей **q** и введём команду

```
block_dump 3240070
```

Утилита ***block_dump*** выведет данные блока по указанному номеру в HEX и ASCII формате, и в них будет видно содержимое текстового файла (или содержимое папки).

```
debugfs:  block_dump 3240070
0000  4865 6c6c 6f2c 2057 6f72 6c64 210a 0000  Hello, World!...
0020  0000 0000 0000 0000 0000 0000 0000 0000  ................
*
```

Так как ASCII кодировка не поддерживает кириллицу, то увидеть слова на русском языке не получится.

Из ***debugfs*** можно выйти, введя символ **q**.

1.6. Ссылки на файлы

Символические («мягкие») и жёсткие ссылки — это файлы, особенность файловой системы Linux, которая позволяет размещать один и тот же файл в нескольких директориях.

Это очень похоже на ярлыки в Windows, так как файл на самом деле остаётся там же где и был, но вы можете на него сослаться из любого другого места.

Символические ссылки более всего похожи на обычные ярлыки. Они содержат адрес нужного файла в вашей файловой системе. Когда вы пытаетесь открыть такую ссылку, то открывается целевой файл или папка. Главное ее отличие от жестких ссылок в том, что при удалении целевого файла ссылка останется, но она будет указывать в никуда, поскольку файла на самом деле больше нет. При этом символическая ссылка является **отдельным файлом**, то есть имеет собственный Inode.

Чтобы создать символическую ссылку на файл, используется команда

```
ln -s <путь до файла> <имя символической ссылки>
```

Например, для файла, лежащего по пути **~/Catalog/file.txt**, создание ссылки выглядит следующим образом:

```
ln -s ~/Catalog/file.txt softlink
```

Тогда в текущей директории появится файл с именем **softlink**. Если прочитать **softlink** командой **cat**, то выведется содержимое файла **file.txt**

```
cat softlink
Hello, World!
```

С помощью команды **ls –l** можно посмотреть, на что ссылается ссылка.

Жёсткая ссылка реализована на более низком уровне файловой системы. В отличии от мягкой ссылки, жёсткая не является отдельным файлом. По факту, это тот же файл, на который и ссылается ссылка, так как имеет такой же Inode.

При создании жёсткой ссылки новый файл не создается, а лишь добавляется запись в каталог, при этом такой атрибут файла, как счётчик ссылок (link count), увеличивается на единицу.

При удалении жёсткой ссылки счётчик ссылок (link count) в индексном дескрипторе уменьшается на единицу. Файл считается удаленным, если счетчик ссылок равен 0.

Создаётся той же командой без параметров

```
ln ~/Catalog/file.txt hardlink
```

Количество жёстких ссылок на файл можно посмотреть командой **ls -l**.

1.7. Утилита dd

Довольно часто системным администраторам приходится копировать различные двоичные данные. Например, иногда может понадобиться сделать резервную копию жесткого диска, создать пустой файл, заполненный нулями для организации пространства подкачки или другой виртуальной файловой системы.

Для решения всех этих задач используется утилита dd linux, которая просто выполняет копирование данных из одного места в другое на двоичном уровне. Она может скопировать CD/DVD диск, раздел на диске или даже целый жесткий диск.

Фактически, это аналог утилиты копирования файлов **cp** только для блочных данных. Утилита просто переносит по одному блоку данных указанного размера с одного места в другое. Поскольку в Linux все, в том числе, устройства, считается файлами, вы можете переносить устройства в файлы и наоборот.

Синтаксис утилиты следующий:

```
dd if=источник_копирования of=место_назначения параметры
```

Существуют дополнительные параметры:

- **bs** - указывает сколько байт читать и записывать за один раз;
- **cbs** - сколько байт нужно записывать за один раз;
- **count** - скопировать указанное количество блоков, размер одного блока указывается в параметре bs;
- **conv** - применить фильтры к потоку данных;
- **ibs** - читать указанное количество байт за раз;
- **obs** - записывать указанное количество байт за раз;
- **seek** - пропустить указанное количество байт в начале устройства для чтения;
- **skip** - пропустить указанное количество байт в начале устройства вывода;
- **status** - указывает насколько подробным нужно сделать вывод;
- **iflag**, **oflag** - позволяет задать дополнительные флаги работы для устройства ввода и вывода, основные из них: nocache, nofollow.

Очень важная и полезная опция - это bs. Она позволяет очень сильно влиять на скорость работы утилиты. Этот параметр позволяет установить размер одного блока при передаче данных. Здесь нужно задать цифровое значение с одним из таких модификаторов формата:

- **c** - один символ;
- **b** - 512 байт;
- **kB** - 1000 байт;
- **K** - 1024 байт;
- **MB** - 1000 килобайт;
- **M** - 1024 килобайт;
- **GB** - 1000 мегабайт;
- **G** - 1024 мегабайт.

Например, мы можем создать файл размером 512 мегабайт, заполнив его нулями из /dev/zero

```
sudo dd if=/dev/zero of=file.img bs=1M count=512
```

1.8. Разметка диска

Во время выполнения различных задач по администрированию системы может понадобится работать с файловой системой Linux, форматировать разделы, изменять их размер, конвертировать файловые системы, выполнить дефрагментацию в Linux или восстановление файловых систем.

Утилита ***mkfs*** позволяет создавать файловые системы на новых дисках. Ее синтаксис выглядит следующим образом:

```
sudo mkfs -t <тип ФС> <устройство (файл)>
```

Например, разметим созданный в предыдущем пункте диск **file.img** с файловой системой ext4. Для этого воспользуемся командой

```
sudo mkfs -t ext4 file.img
```

1.9. Монтирование диска

Монтирование в Linux — это подключение в один из каталогов целой файловой системы, которая находится на другом устройстве.

Эту операцию можно представить как «прививание» ветки к дереву. Для монтирования необходим пустой каталог — он называется точкой монтирования. Точкой монтирования может служить любой каталог.

Для монтирования в Linux используется команда **mount** со следующим синтаксисом:

```
mount файл_устройства папка_назначения
```

Например, примонтируем размеченный диск **file.img** в папку **Dir**.

```
sudo mount file.img ~/Dir
```

Чтобы проверить, куда примонтировался диск и какая у него файловая система, можно командой **df -T**.

После перезагрузки системы связь с примонтированными дисками теряется. Чтобы сохранить примонтированный диск и не повторять операцию после каждого перезапуска необходимо добавить строчку в файл /etc/fstab.

В общем виде содержимое файла представлено в виде строк:

```
раздел_диска  точка_монтирования   файловая_система      опции_монтирования    два_числа
```

Например:

```
/dev/sda   /mvt/crdom0    ext4       default     0 0
```

Опции монтирования:

- **defaults** Использовать настройки по умолчанию (rw,suid,dev,exec,auto,nouser,async)
- **rw** / **ro** — Разрешено чтение и запись / Разрешено только чтение
- **suid** / **nosuid** — Разрешение / Блокировка работы suid, и sgid бит
- **dev** / **nodev** — Интерпретировать / не интерпретировать блок специальных устройств на файловой системе.
- **exec** / **noexec** — Разрешить выполнять двоичные файлы находящиеся на этом диске / Запретить

- **auto** / **noauto** — Устройство будет устанавливаться автоматически при загрузке / Не будет
- **nouser** / **user** — Запрещение монтирование от всех кроме root (nouser) / Разрешение монтировать от лица любого пользователя
- **async** / **sync** — Запись и чтение на диске будут производиться асинхронно / Синхронно

Вспомогательные числа

- **Первое число** — возможные значения 0 или 1 — означает, включить/выключить резервное копирование файловой системы при помощи команды dump. Устаревшая опция.
- **Второе число** — возможные значения 0, 1, 2, — означает порядок, в котором файловая система должна быть проверена при загрузке:
    - 0 — не проверять.
    - 1 — должна проверяться первой и использоваться как корневая.
    - Для всех остальных систем ставится 2.

Для размонтирования используется команда **umount**.

1.10. Архивация файлов

В качестве инструмента для архивации данных в Linux используются разные программы. Например архиватор Zip , приобретший большую популярность из-за совместимости с ОС Windows. Но это не стандартная для системы программа. Поэтому хотелось бы осветить команду **tar** — встроенный архиватор.

Изначально tar использовалась для архивации данных на ленточных устройствах. Но также она позволяет записывать вывод в файл, и этот способ стал широко применяться в Linux по своему назначению. Чтобы создать новый архив, в терминале используется следующая конструкция:

```
tar опции архив.tar <файлы_для_архивации>
```

С помощью следующей команды создается архив **archive.tar**, включающий файлы file1, file2 и file3:

```
tar -c -f archive.tar file1 file2 file3
```

Следующая команда выводит содержимое архива, не распаковывая его:

```
tar -tf archive.tar
```

Следующая команда распаковывает архив:

```
tar -xvf archive.tar
```

Важно отметить, что архивирование в Linux это не одно и то же, что и сжатие файлов. Архивирование — это объединение нескольких небольших файлов в один, с целью более удобной последующей передачи, хранения, шифрования или сжатия. Для сжатия часто используется утилита ***gzip*** со следующим синтаксисом:

```
gzip опции файл
```

Чтобы сжать папку в Linux вам придется сначала заархивировать ее с помощью tar, а уже потом сжать файл архива с помощью gzip.

```
gzip archive.tar
```

Распаковка производится командой

```
gunzip archive.tar
```

Лицезреть разницу в размерах сжатого и несжатого архивов можно командой

```
ls -lh
```

2. Практическая часть

2.1. Задание 1

2.1.1. Создайте в домашней директории каталог **Lab_2** и перейдите в неё.

2.1.2. Оставаясь в директории **Lab_2**, создайте в каталоге **books** текстовый файл, имеющий название вашего любимого фильма/мультфильма и содержащий год выхода произведения, название студии или имя режиссёра (на английском языке).

2.1.3. Перенесите созданный файл из директории **books** в **good_films**.

2.1.4. Удалите каталог **books**.

2.1.5. Создайте копию директории **good_films** и поместите её в домашнюю директорию.

2.1.6. Переименуйте директорию **good_films** в **bad_films**. Убедитесь, что внутри остался файл с вашим фильмом.

2.1.7. Отредактируйте файл в **bad_films** так, чтобы он содержал аналогичную информацию о фильме, который вас разочаровал.

2.2. Задание 2

2.2.1. Выясните номер Inode файла, лежащего в папке **good_films**.

2.2.2. Прочитайте содержимое файла по номеру Inode. Подсказка, *используйте утилиту debugfs*.

2.2.3. Создайте каталог **good_music.** Заполните его тремя файлами, название которых совпадает с названием ваших любимых музыкальных композиций, содержимое – текст песни (при наличии), автор и его дата рождения.

2.2.4. Определите номер Inode созданной директории и всех содержащихся в ней файлов. Прочитайте содержимое и содержимое каталога по их номерам Inode.

2.2.5. Удалите любой из файлов и попробуйте прочитать его. Возможно ли это сделать? Изменилось ли содержимое файла каталога?

2.3. Задание 3

2.3.1. Создайте на файл, лежащего в папке **bad_films**, две ссылки – жёсткую и символическую. Убедитесь, что они работают.

2.3.2. Переместите файл в директорию **good_films** и проверьте работоспособность ссылок.

2.3.3. Удалите перемещенный файл и аналогично проверьте работу ссылок.

2.4. Задание 4

2.4.1. Создайте файл, содержащий нули объемом 100 KiB. Запись производить блоками по 512 байт. Подсказка, *необходимо использовать утилиту dd.*

2.4.2. Разметьте его с файловой системой fat32 (*при создании укажите тип vfat*).

2.4.3. Создайте каталог **new_dir** и смонтируйте в него созданную файловую систему.

2.4.4. Убедитесь, что файловая система смонтирована корректно (запишите в неё и прочитайте несколько файлов).

2.5. Задание 5

Необходимо использовать утилиты tar, gzip

2.5.1. Добавьте файлы из примонтированного каталога **new_dir** в архив.

2.5.2. Добавьте в архив со сжатием все текстовые файлы. Сравните объемы данных.

2.5.3. Распакуйте созданные архивы.

Контрольные вопросы

1. Что такое файловая система? Какие существуют типы?

2. Что такое Inode?

3. В чём разница между символической ссылкой и жёсткой ссылкой?

4. Что такое каталог с точки зрения ядра Linux?

5. В чём разница между tar и gzip?