Окружение

Окружение – некоторое информационное пространство, в котором можно задать именованные хранилища данных (**переменные окружения**), а в последствии хранить в них информацию и использовать её.

Переменные окружения – именованные переменные, содержащие текстовую информацию, которую могут использовать запускаемые программы. Значением такой переменной может быть, например, место размещения исполняемых файлов в системе, имя предпочитаемого текстового редактора или настройки системной локали. Посмотреть установленные переменные можно командой **env** (*environment*) или **printenv**.

Важные переменные:

- **PATH**

  Определяет список директорий, в которых операционная система будет искать исполняемые файлы. Когда обычная команда, например, **ls** интерпретируется командной оболочкой (такой как bash), оболочка ищет исполняемый файл с указанным именем в этом списке, и, если находит, запускает файл, передав ему указанные аргументы командной строки. Чтобы запускать исполняемые файлы, пути к которым не находятся в PATH, необходимо указывать полный путь к файлу, например /bin/**ls.**

- **HOME**

  Содержит путь к домашней директории текущего пользователя. Эта переменная может использоваться приложениями для определения расположения файлов настроек пользователя, который их запускает.

- **PWD**

  Содержит путь к рабочему каталогу.

- **OLDPWD**

  Содержит путь к предыдущему рабочему каталогу, то есть, значение PWD перед последним вызовом cd.

- **SHELL**

  Содержит имя текущей командной оболочки, например, bash.

- **TERM**

  Содержит имя запущенного терминала, например xterm (xterm-256color – терминал xterm с поддержкой 256 цветов)

- **PAGER**

Указывает команду для запуска программы постраничного просмотра содержимого текстовых файлов, например, /bin/less.

- **EDITOR**

  Содержит команду для запуска программы для редактирования текстовых файлов, например /usr/bin/nano.

- **MANPATH**

  Содержит список каталогов, которые использует man для поиска man-страниц. Стандартным значением является /usr/share/man:/usr/local/share/man

- **TZ**

  Может использоваться для установки временной зоны (например: время в Лондоне). Доступные временные зоны можно найти в /usr/share/zoneinfo/

- LC_ALL=en_US.UTF-8

  Устанавливает локаль для всех категорий в значение en_US.UTF-8, что обеспечивает корректное отображение символов в Unicode.

- USER=kirill.m

  Содержит имя текущего пользователя.

- LANG=en_US.UTF-8

  Устанавливает языковые настройки, в данном случае, на английский (en) с кодировкой UTF-8

  Следующие файлы следует использовать для установки переменных окружения на уровне системы: /etc/profile, /etc/bash.bashrc и /etc/environment.

- /etc/profile устанавливает переменные только для командных оболочек. Он может запускать любые скрипты в оболочках, совместимых с Bourne shell.
- /etc/bash.bashrc устанавливает переменные только для интерактивных оболочек. Он также запускает bash-скрипты.
- /etc/environment используется модулем PAM-env. Здесь можно указывать только пары имя=значение.

На уровне пользователя

Вам не всегда нужно будет устанавливать переменные окружения на уровне системы. Например, вы можете добавить ваш каталог /home/пользователь/bin в PATH, , однако, если не хотите, чтобы это затрагивало других пользователей системы. Переменные окружения пользователя можно устанавливать во многих других файлах

Пайпы (конвейеры)

Символ | в командной строке Linux (и в других Unix-подобных системах) обозначает "пайп" (или "конвейер"). Этот символ используется для передачи вывода одной команды в качестве ввода для другой команды.

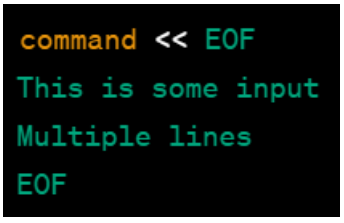
Команды - фильтры

Команды-фильтры работают с потоками данных, принимая информацию из стандартного ввода и отправляя результат в стандартный вывод. Это позволяет использовать их в цепочках с другими командами, объединяя их в мощные конструкции для обработки данных.

1. **grep**
   Команда grep используется для поиска строк, соответствующих заданному шаблону в тексте.
2. **sort**
   Команда sort используется для сортировки строк в потоке данных.
3. **head**
   Команда head используется для вывода начальных строк из файла или потока данных.
4. **tail**
   Команда tail используется для вывода конечных строк из файла или потока данных.

Перенаправления вывода и ошибок.

1. **Перенаправление вывода (>)**
   Используется для перенаправления стандартного вывода (stdout) команды в файл. Если файл существует, он будет перезаписан. Пример: **command > file.txt**
2. **Добавление в файл (>>)**
   Используется для добавления вывода команды в конец существующего файла, вместо перезаписи файла. Пример: **command >> file.txt**
3. **Перенаправление ошибок (2>)**
   Используется для перенаправления стандартного вывода ошибок (stderr) команды в файл. Пример: **command 2> error.txt**
4. **Перенаправление вывода ошибок в один файл (&> или >&)**
   Используется для перенаправления и стандартного вывода, и стандартного вывода ошибок в один файл. Пример: **command &> output_and_error.txt**
5. **Перенаправление ввода (<)**
   Используется для перенаправления ввода из файла в команду. Пример: **command < input.txt**
6. **Here Document или встроенный документ (<<)**
   Используется для передачи блока текста в качестве ввода для команды.
   Пример:



Выдача прав обычному пользователю (root/sudo)

1. **root**
   Системная учетная запись в Unix-подобных операционных системах, которая обладает полными (или почти полными) привилегиями на уровне всей системы. В терминологии Unix и Linux, root также называется суперпользователь или администратор системы.
2. **sudo**
   Команда sudo используется для предоставления временных привилегий суперпользователя обычному пользователю. Обычно, чтобы выполнить команду от имени суперпользователя, вы можете использовать: **sudo command**

Терминал, консоль и т. д.

Терминал (text input/output environment) – это интерфейс, предоставляющий пользователю текстовую среду для взаимодействия с операционной системой.

Консоль (physical terminal) – это текстовый интерфейс, предоставляющий пользователю доступ к командам операционной системы.

Командная оболочка - программа, которая обеспечивает прямую связь между пользователем и ОС. В командной оболочке программы выполняются, и результат выполнения отображается на экране.

Сценарий командной оболочки - текстовый файл, содержащий последовательность команд, которые обычно выполняются в командной оболочке.

Shell (command line interpreter) – командная оболочка, программа, предоставляющая интерфейс между пользователем и операционной системой.

Командная строка - это текстовый интерфейс взаимодействия с операционной системой, в котором пользователь может вводить команды для выполнения различных задач.

Интерпретатор - программа, которая читает и выполняет исходный код программы построчно или по блокам, не компилируя его предварительно в машинный код.

Inode

inode - это структура данных в файловых системах Unix и Linux, предназначенная для хранения метаданных (*размер файла, права доступа к файлу, владелец файла, идентификатор группы файла, дата и время создания, дата последней модификации, тип файла*) файла. У каждого файла или каталога есть свой уникальный inode, благодаря которому возможно однозначно идентифицировать файл или каталог. Inode Хранит адреса блоков данных, где фактически хранится содержимое файла. Inode Хранит адреса блоков данных, где фактически хранится содержимое файла.

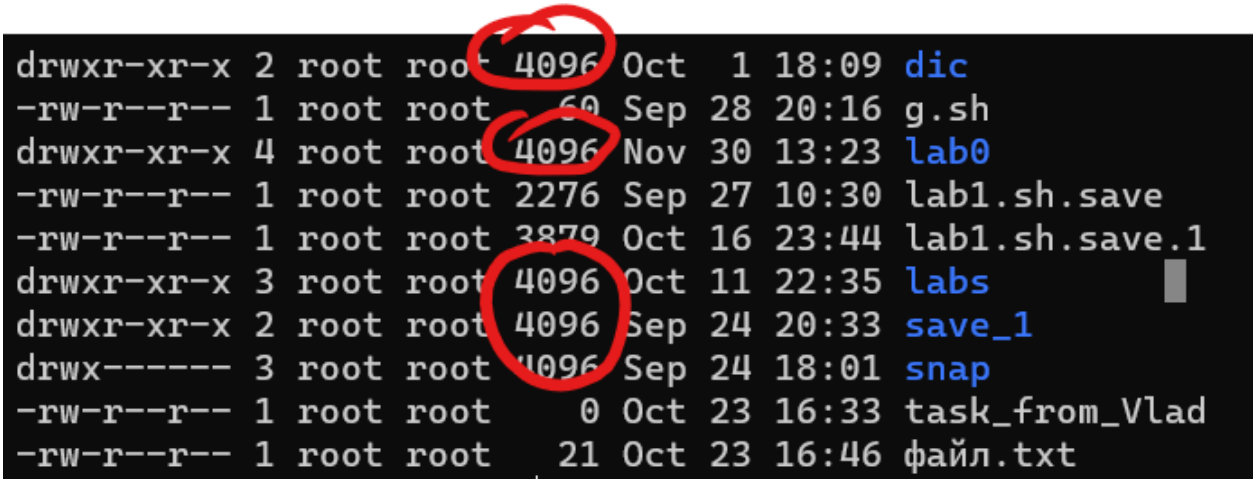
Команды:

- **df -i** Команда, предназначенная для просмотра всех доступных inode (их кол-во в системе ограничено).

- **ls -i (*Имя файла/директории*)** Команда, позволяющая узнать inode для конкретного файла или директории.

Как защищался я:

Я сказал, что Inode ссылается на 4 блока. Объём занимаемой памяти одного такого блока можно рассчитать. Пишем команду ls -l и ищем часто попадающийся объём памяти. Далее делим его на 4 и находим объём памяти, занимаемый одним блоком.



Устройство памяти в ЭВМ.

Память – это устройство, хранящее команды и данные. Это устройство состоит из одинаковых по размеру блоков, которые называются *ячейками памяти*. Каждая ячейка памяти хранит одно слово информации (*определенное количество бит, равное размеру ячейки данных*). Слово информации состоит из элементов памяти, каждый бит = 1 элементу памяти. Очевидно, что каждый элемент памяти равен либо 0, либо 1, поскольку так устроена ЭВМ. И совокупность всех этих элементов данных представляет собой слово информации = ячейка данных.

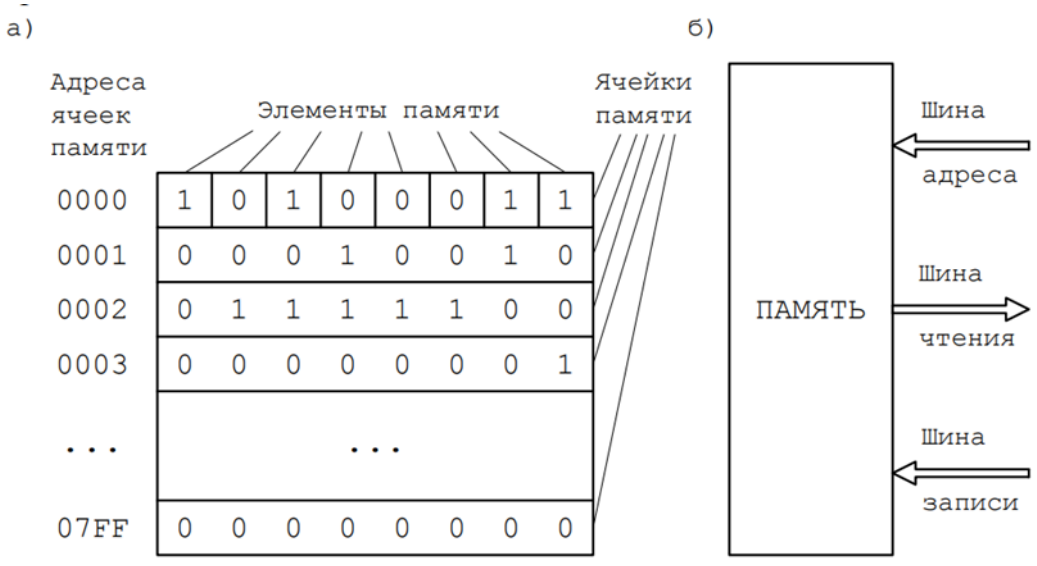


Рис. 1.7. Организация памяти ЭВМ

Каждая ячейка нумеруется определенным числом – адресом ячейки памяти. Если необходимо записать в память слово, следует подать на шину адреса памяти сигналы, соответствующие адресу ячейки, в которую надо поместить записываемое слово, и подать

само слово на шину записи. Память устроена так, что заданное слово будет передано в ячейку с указанным адресом и может храниться там сколь угодно долго. В любой момент, обратившись к памяти, можно получить содержимое хранимого там слова.

На рис. 1.7а показана память, имеющая 4096 = 218-разрядных слов, т. е. содержащая 4096 байт информации. При том же самом числе запоминающих элементов можно было бы организовать память из 32 768 1-битовых слов, 2048 16-битовых слов, 1024 32-битовых слов и т. д. Если нужно обрабатывать информацию, которая может кодироваться лишь двумя символами (например, некоторые данные переписи населения: пол, семейное положение и т. п.), то выгодно использовать первую организацию памяти. В случае же точных вычислений приемлемее память из 32-битовых или даже 64-битовых слов. Однако при выборе разрядности ячеек памяти надо учитывать, что в них должны храниться и команды программы, используемые процессором для обработки таких данных. А как же строится команда ЭВМ и какова ее разрядность?

Виды памяти

- **Физическая память (Physical Memory)**
  Фактическая аппаратная память, установленная на компьютере. Информацию о физической памяти можно получить с помощью команды free или cat /proc/meminfo.
- **Виртуальная память (Virtual Memory)**
  Косвенная абстракция, предоставляемая операционной системой, которая позволяет приложениям получать доступ к большему объему памяти, чем физически доступно. Виртуальная память включает в себя как оперативную, так и подкачиваемую память (swap). В случае нехватки оперативной памяти, Linux может использовать часть жесткого диска для временного хранения данных, переносимых в и из оперативной памяти.

  MMU (блок управления памятью) преобразует адреса физической памяти в виртуальные. Эти адреса не зависят от того, где они расположены. Эти адресные пространства известны как “страницы”. Вместе все они образуют адресное пространство.

  Страницы - это блоки памяти фиксированного размера. Они служат для эффективного управления и выделения памяти в операционной системе.

Иерархия памяти (сверху вниз по скорости (сверху самая быстрая)):

1. **Регистры**
   Они встроены в сам процессор и доступ к ним возможен практически мгновенно.
2. **Кэш-память**
   Небольшой объём памяти, расположенный в ЦП, в котором хранятся часто используемые данные для повышения производительности.
3. **ОЗУ (RAM)**
   Представляет собой основную память, используемую системой для хранения данных и программ, которые используются в данный момент. ОЗУ намного медленнее регистров и кэш-памяти, но имеет гораздо большую емкость. Кроме того, оперативная память энергозависима, а это означает, что ее содержимое теряется при выключении системы.
4. **Пространство подкачки**
   Часть жесткого диска, в которой хранятся данные, которые в данный момент не используются в оперативной памяти. Когда в системе заканчивается доступная оперативная память, она перемещает часть наименее используемых данных в пространство подкачки, чтобы освободить место для более важных данных. Однако, хранение данных на жёстком диске значительно замедляет доступ к ним, а следовательно, и производительность системы.

Также существует **файл подкачки**

Файл подкачки предоставляет альтернативный метод реализации подкачки без необходимости выделения отдельного раздела. Вместо этого он представляет собой обычный файл, создаваемый в файловой системе.

Для чего нужна *виртуальная память*?

Благодаря виртуальной памяти можно запускать большие программы, т. к. объём виртуальной памяти гораздо больше физической. Однако виртуальная память уступает в скорости физической. Это является её самым главным недостатком.

Сегментно-страничная организация памяти

Сегмент - логический блок адресного пространства программы, который содержит определенный тип данных или выполняет определенную функцию. Сегментация является одним из методов организации виртуальной памяти.

Сегментация - разбиение адресного пространства на сегменты, такие как код, данные, стек и др.

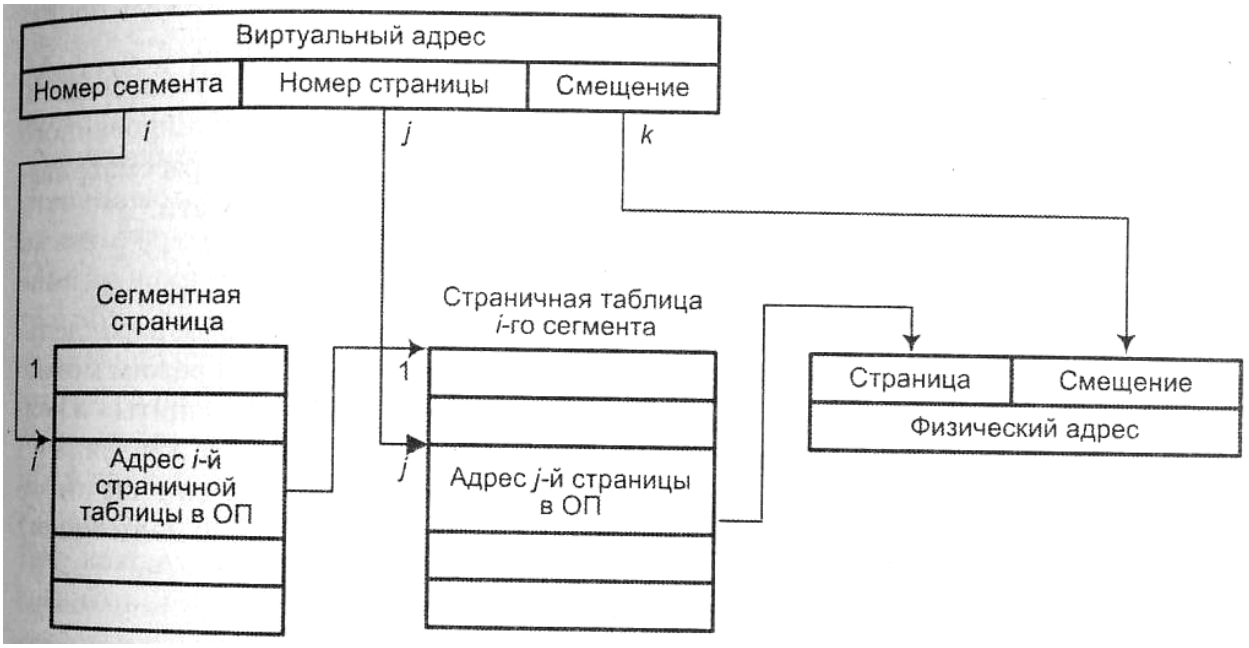
Страничной организация *виртуальной памяти* — непрерывный массив со сквозной нумерацией слов, что не всегда можно признать оптимальным. Обычно программа состоит из нескольких частей — *кодовой*, *информационной* и *стековой*. Так как заранее неизвестны длины этих составляющих, то удобно, чтобы при программировании каждая из них имела собственную нумерацию слов, отсчитываемых с нуля. Для этого организуют систему *сегментированной памяти*. В каждом сегменте устанавливается своя собственная нумерация слов, начиная с нуля. Виртуальная память также разбивается на сегменты, с независимой адресацией слов внутри сегмента. Каждой составляющей программы выделяется сегмент памяти. Виртуальный адрес определяется номером сегмента и адресом внутри сегмента. Для преобразования виртуального адреса в физический используется специальная сегментная таблица.

Недостатком такого подхода является то, что неодинаковый размер сегментов приводит к неэффективному использованию ОП. Так, если ОП заполнена, то при замещении одного из сегментов требуется вытеснить такой, размер которого равен или больше размера нового. При многократном повторе подобных действии в ОП остается множество свободных участков, недостаточных по размеру для загрузки полного сегмента.

Решением проблемы: сегментно-страничная организация памяти.

Размер *сегмента* выбирается не произвольно, а задается кратным размеру *страницы*. Сегмент может содержать то или иное, но обязательно целое число страниц, даже если одна из страниц заполнена частично. Возникает определенная иерархия в организации доступа к данным, состоящая из трех ступеней: сегмент > страница > слово. Этой структуре соответствует иерархия таблица служащих для перевода виртуальных адресов в физические. В сегментной таблице программы перечисляются все сегменты данной программы с указанием начальных адресов СТ, относящихся к каждому сегменту. Количество страничных таблиц равно числу сегментов и любая из них определяет

расположение каждой из страниц сегмента в памяти, которые могут располагаться не подряд — часть страниц может находиться в ОП, остальные — во внешней памяти.



LVM

Logical Volume Management (управление логическими томами) - дополнительный слой абстракции от железа, позволяющий собрать кучи разнородных дисков в один, и затем снова разбить этот один именно так как нам хочется. LVM обеспечивает эффективное и гибкое управление хранилищем данных.

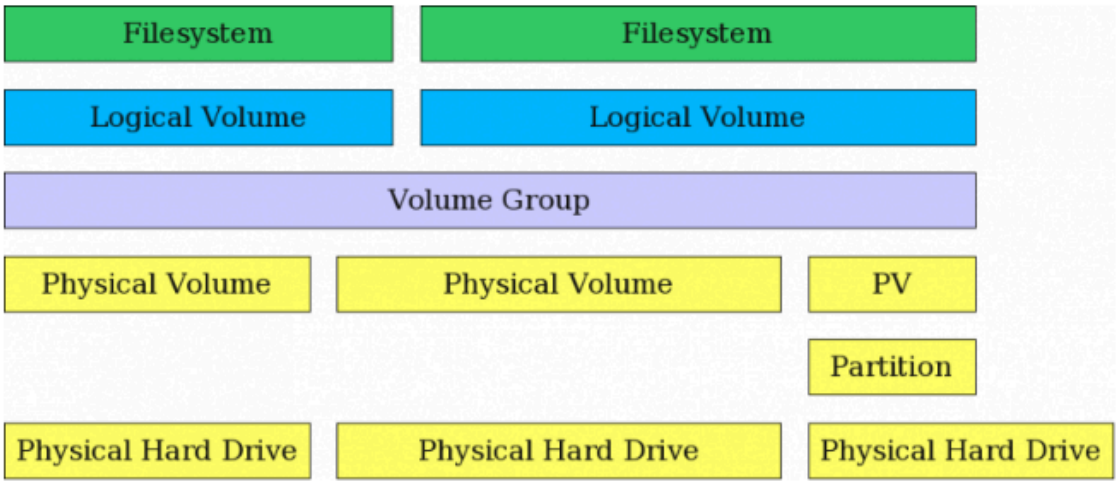


Рисунок 1. LVM позволяет объединять разделы и целые жесткие диски в группы томов.

3 уровня абстракции:

1. **PV (Physical Volume)** — физический том - абстрактное блочное устройство. Физические тома хранят значения жёстких дисков или их разделов (партиций).

2. **VG (Volume Group)** — группа томов - абстрактное блочное устройство, предназначенное для объединения физических томов ***PV*** в группу. Создаётся единый диск, который будет дальше разбиваться так, как нам хочется.

3. **LV (Logical Volume)** — логический раздел, собственно раздел нашего нового «единого диска» (Группы Томов), который мы потом форматируем и используем как обычный раздел, обычного жёсткого диска.

Партиции (Partitions)

В компьютерных системах жесткий диск может быть разделен на логические части, называемые партициями. Каждая партиция представляет собой отдельный блок пространства на диске, который можно форматировать и использовать для хранения данных. Партиции могут содержать файловые системы и служить различными целями, такими как установка операционной системы, хранение данных и т. д.

Места хранения некоторых важных данных

1. **/etc/passwd**
   Файл /etc/passwd содержит информацию о пользователях, такую как их имена, уникальные идентификаторы (UID), идентификаторы групп (GID), полные имена и др. Однако, в современных системах, пароли теперь обычно хранятся в файле /etc/shadow.
2. **/etc/shadow**
   Файл /etc/shadow хранит зашифрованные пароли пользователей и другую чувствительную информацию, такую как политики паролей, даты последнего изменения пароля и т. д. Этот файл обычно доступен только суперпользователю (root) для обеспечения безопасности.
3. **/etc/group**
   Файл /etc/group содержит информацию о группах пользователей, включая их идентификаторы (GID) и членов группы.
4. **/etc/shells**
   Файл /etc/shells содержит пути к исполняемым оболочкам (shells) в системе. Этот файл используется, чтобы определить, какие оболочки могут быть использованы пользователями при установке своей оболочки по умолчанию.
5. **/etc/gshadow**
   Файл /etc/gshadow аналогичен файлу /etc/shadow, но для групп. Он содержит информацию о зашифрованных паролях групп.

6. **/etc/sudoers**
   Файл /etc/sudoers содержит настройки для команды sudo, которая позволяет пользователям выполнить команды с привилегиями суперпользователя. Редактирование этого файла требует особых прав.

Конченые вопросы

Как сама древовидная структура файлов располагается в памяти, если файловых систем некоторое количество? **Ответ:** Если в системе присутствует несколько файловых систем, каждая из них имеет свою собственную древовидную структуру файлов и каталогов. Операционная система обычно поддерживает множество файловых систем, и каждая из них может быть монтирована в определенной точке в единой древовидной иерархии.

Источники:

- https://ru.hexlet.io/courses/cli-basics/lessons/environment-variables/theory_unit
- https://journal.sweb.ru/article/shpargalka-bazovye-komandy-dlya-terminala-linux#:~:text=%D0%A2%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B0%D0%BB%20%D0%9B%D0%B8%D0%BD%D1%83%D0%BA%D1%81%20(Linux%20Terminal)%20%E2%80%94,%D1%81%20%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%BE%D0%B9%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BE%D0%B9%20%D0%B8%20%D0%B4%D1%80.
- https://www.baeldung.com/linux/user-memory-usage#:~:text=In%20Linux%2C%20there%20are%20two,memory%20installed%20on%20your%20computer.
- https://opensource.com/business/16/9/linux-users-guide-lvm
- https://www.geeksforgeeks.org/paged-segmentation-and-segmented-paging/