**ВО ВСЕХ ПРИМЕРАХ НИЖЕ ИСПОЛЬЗОВАЛАСЬ ОБОЛОЧКА - BASH**

**Терминал** - графическая программа эмулирующая консоль. Такие программы позволяют не выходя из графического режима выполнять команды.

Терминал по сравнению с консолью имеет дополнительный функционал (управление мышью, контекстное меню, полоса прокрутки, вкладки, запуск нескольких окон, главное меню, графические настройки). Вызвать терминал можно с помощью клавиш ctrl+alt+t

## **Программная оболочка**

Консоль и терминал обрабатывают команды с помощью программной оболочки.

Программная оболочка - интерпретатор команд, он распознает команды введенные в командной строке, и запускает программы для выполнения команды.(НАВЕРНО КАЖДАЯ ОБОЛОЧКА ДЕЛАЕТ ЭТО ПО СВОЕМУ, ХОТЯ И СХОЖА)

В [Ubuntu](https://help.ubuntu.ru/wiki/ubuntu) по умолчанию используется оболочка [bash](https://help.ubuntu.ru/wiki/bash), он распознает команды на языке bash.

Bash можно заменить на другую оболочку, их существует несколько. Каждая оболочка имеет свой набор настроек и возможностей.

(автовыполнение команд при входе в оболочку, внутренние команды оболочки, ведение истории, можно назначать сокращенные команды - алиасы).

**Команды**

Команды - это предопределенный набор букв, цифр, символов, которые можно ввести в командной строке и выполнить нажав энтер.

Команды делятся на два вида:

команды встроенные в программную оболочку (например history)

команды управляющие программами, установленными в системе

Команды для управления программами строятся по такой схеме: название\_программы -ключ значение

**Л**юбая команда в терминале. это установленная в оперативку программа, если такая программа есть комп её выполнит. если нет он тупо не знает этой программы. Где хранятся программы операционной системы? Для этого есть команда:

**echo $PATH** - выводит что-то вроде: **/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin**

где каждое новое двоеточие обозначает новую папку. а вся строка соответственно путь к этим программам/командам операционной системы.

**which название\_команды -** вернёт местоположение конкретной программы

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Синтаксис построения команд**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Команда** состоит из набора сущностей и формируется следующим образом: **имя\_команды опции(ключи) операнд/аргумент/значение**

также возможна вариация когда указывается например только название программы. Например: **date** - что она делает понятно но сам синтаксис включает в себя только название и всё, никаких опций и/или значений. А теперь такая-же команда но с опцией(ключом) **date -u** -она выводит universal time (utc). Но синтаксис здесь: название\_команды опция . Если бы у date было несколько опций (ключей) и надо было бы их разом вызвать то я бы писал что-то типа : **date -u -a -c,** где u a c - это опции а то как они записаны (с пробелами и дефисами) это синтаксис вызывания какой то команды с несколькими опциями. При этом допускается date -uac или **date -auc** также отработало бы корректно (при использовании коротких опций их можно склеивать друг к другу). у каждой программы свой набор ключей (опций), они перечислены в справке к программе, ключи используются для указания какие настройки использовать или какое действие выполнить.

**Значение(операнд, аргумент)** - адрес, цифры, текст, спецсимволы (\*, ~, \, &, « », \_ ), переменные ($HOME, $USER, $PATH).

Существуют расширенные опции и пишутся они: date –option –option2 –option3, т.е. перед каждой опцией стоит два дефиса.

Теперь пример команды без опции, но с значением (каким-то input-ом/аргументом/операндом):

**echo hello** - input или значение ещё называют **операндом** (здесь это hello), может быть одно имя и несколько операндов (значений), или одно имя и несколько опций, или имя -опция её\_значение -опция её\_значение, вариаций много.

**whoami** - команда возвращающая текущего пользователя

ctrl+alt+t - горячие клавиши для вызова терминала.

**lsb\_release -a** - посмотреть версию ubuntu

**exit** - выход/закрытие из терминала

**echo** **Текст\_который\_хочеш\_вывести\_на\_экран** - выводит текст в консоль но лучше, брать в кавычки то что выводиш так-как может неправильно понять и перекинет в редактор, из редактора выходить через **ctrl+c**

**date** - выводит дату / время

**cal -** показывает календарь (у меня не сработала)

**clear** - очищает терминал от всего что в нем было или нажми **ctrl + l**- это тоже-самое

**history** -показывает историю вызова всех команд в рамках текущего окна с момента включения/ все комаанды выводятся в списке и имеют порядковый номер/ используя этот номер можно вызвать любую команду из списка не прописывая её опять по новому/ Например: **!4** - (выполнит четвёртую команду, пробел не ставить)

**history -c; history -w** - очистить историю (после этого стрелкой вверх уже не получится вызвать предыдущие команды)

**date + %D** - только дата

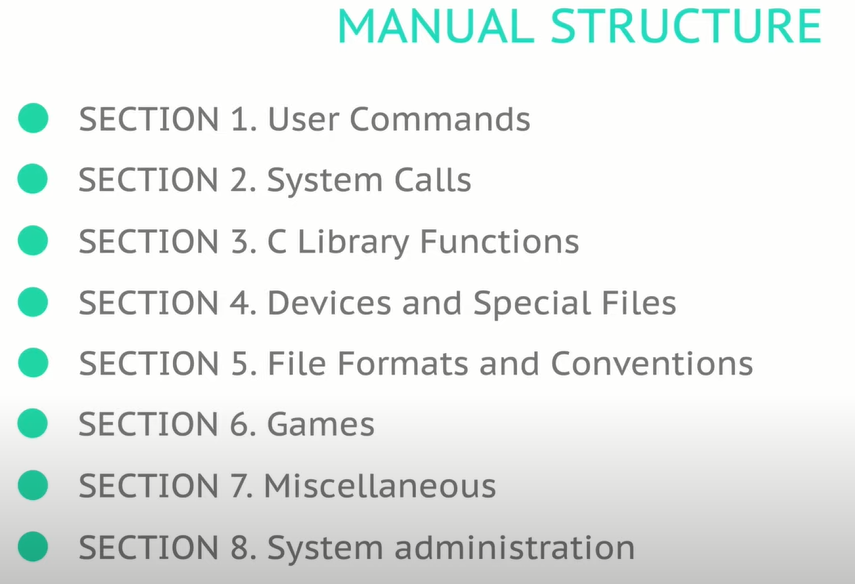
**echo "Hi bro, today is: $(date +%D)"**- выведет -> Hi bro, today is: 04/09/23 то есть знак доллара как бы делает включение какой-то команды которая уже имеется в терминале это не точно буду смотреть дальше

**my\_name=Dimon -** создание переменной / оказывается так можно

**echo "My name is $my\_name”** выводит ->My name is Dimon опять же знак доллара позволяет делать включение в частности здесь это переменная my\_name

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Работа с мануалом**

Как видно есть разделение на части, найболее используемые это : 1, 8

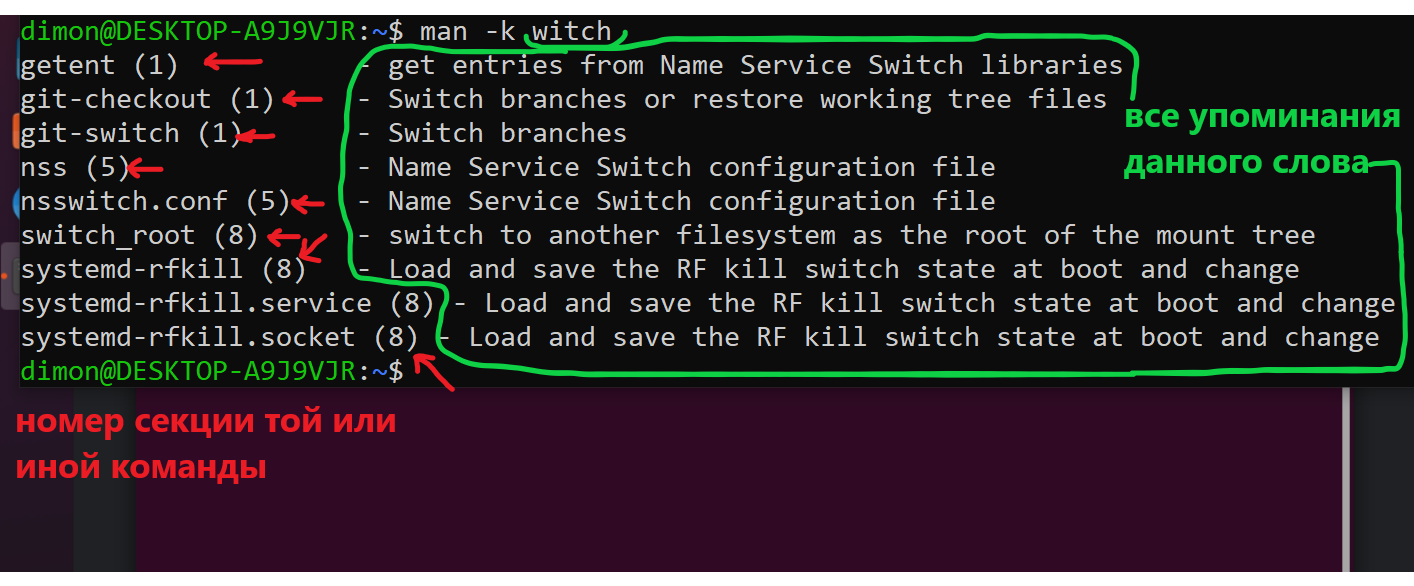
**man** - комана которая направляет в мануал той команды которая указана как значение. Напримр:

**man which**- перенаправит в мануал с описанием команды which

**man man** - покажет мануал по команде man

У данной команды также есть опции и значения. Смотри в мануале (man man)

**man-** сокращённо от manual, легко запомнить

**man -k which** - поиск команды witch, т.е. нам вернётся список где встречается данная команда или слово, на фото Видно, что с права это все упоминания данного слова, а с лева команды в описании которых встречается данное слово, или сама команда-одноимённая.

После того как узнали свой подраздел можно просмотреть мануал конкретной команды где встречается для нас найболее интересный/подходящий вариант. Нас интересует команда witch (1)

**man 1** **which -** вернёт мануал по данной команде, бля но можно и просто **man which-** тот же результат.

Далее идёт пример вызова данной команды и разбор отображаемой информации в мануале.

**[item]** - Квадратные скобки обозначают, что данные опции используются по выбору (опционально) и не являются обязательными.

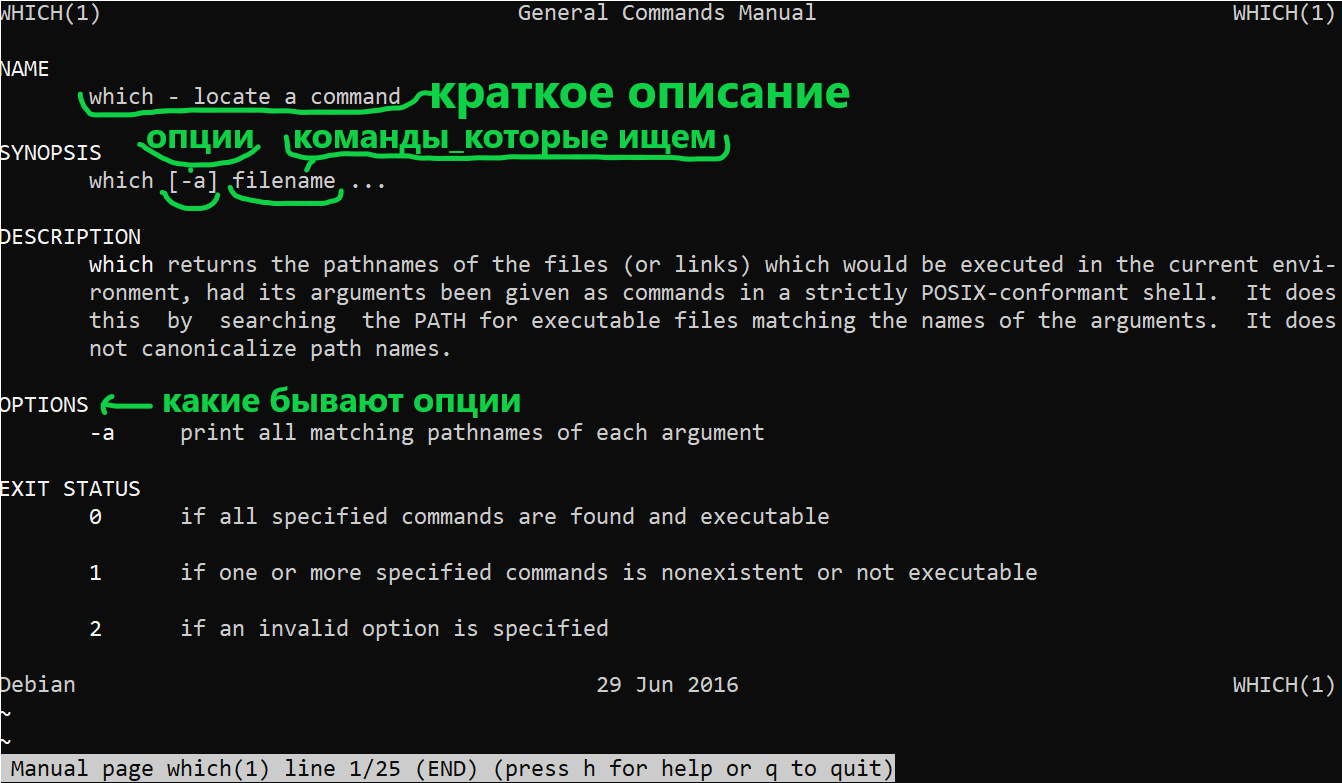
**< item>** - обозначает обязательное использование элемента.

**item …** Три точки в описании команды означает что можно использовать сколько хочешь операндов.

Например: **which date cal ls ←** выведет месторасположение этих трёх команд, по очереди на каждой строке.

**item1 | item2** - использование одного из двух ( это или это)

**SYNOPSIS** - описание как испол



В мануале не работает колёсико мышки, **E** - скролить вниз, **Y** - скролить вверх

**ls** - выводит содержимое папки HOME но только в UBUNTU, на винде ничего не выводит.

**Мануал это круто но можно использовать и Google )))**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**input & output**

**Примеры работы по умолчанию:**

**input** - какие то входные данные

a)- standard input(0) - это обычный ввод с клавы

b)- Command inputs - это те команды которые мы прописываем в терминале

**output** - выходные данные, т.е. то, что мы видим как результат своей работы

a)- standard output(1)

b)-standard error(2)

standard input(0), 

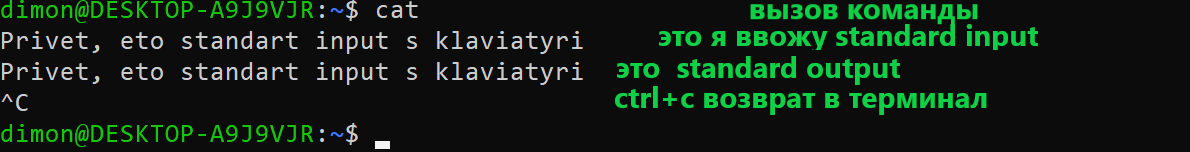
standard Output(1)

standard Error(2)

эти 3 штуки называются **Standard Data Stream**

Значение имеют и цифры: 0-standard input, 1-standard output, 2-standard Error. Это будет использоваться далее в опциях.

**Standard Input -** для демонстрации будет использоваться команда **cat,** у которой, одной из функцией, если не передан файл откуда считывать инф. она будет считывать с терминала, предварительно зайдя в свой какой то редактор, выйти из которого ctrl+c.



**Command Inputs** - или ещё называют Command arguments- это предопределенные значения к командам(те которые запрограммированы), те самые значения или ещё их называют операндами. Например: **cal 5 2023** - здесь 5 и 2023 это и есть значения (в мануале они называются **synopsis**) - эта штуку не является частью data stream-a и её нельзя ни редиректить ни пампить.

**Standard output** - это стандартный вывод на экран, по умолчанию в терминал. Например: **date** выведет текущую дату, это и есть Standard output

**Standard Error** - например: **date dfsdfsdkfowf** т.е. команда с несуществующим значением (операндом), в терминале выводится сообщение об ошибке - вот это сообщение и есть Standard Error.

**Примеры работы с перенаправлением:** по умолчанию всё выводится в терминал, но можно перенаправить чтобы вывод был например в файл. **Перенаправление outputa достигается указанием стрелочек >> или >**.

**Примеры Перенаправление outputa**

**cat** - команда отвечающая за standard input - standard output а также объединяет файлы… хз.

—--------------------------------------- **ввод с терминала в новый файл**—------------------------------------------------------------

**cat 1>** **название\_файла**- обозночает что мы будем делать перенаправление для Standard output, значок **> -** обозначает именно создание нового файла. так как 1-это Standard output, если щас нажать ентер, попадём в редактор ввода текста. Когда ввёл текст, нажал энтер то где-то в недрах users/user/… создаётся данный файл и ввод текста осуществляется туда. Если выполнить команду дважды то содержимое файла будет переписано. Выйти из редактора Ctrl+c., если такого файла нет то он будет создан

—----------------------------------**ввод с терминала в уже существующий файл**—---------------------------------

**cat 1>> название\_файла**- содержимое файла не переписывается, а добавляется, ввод с терминала

—--------**перенаправление\_outputa\_как\_результата\_выполнения\_команды\_в\_файл**---------------

**название\_команды 1>> название\_файла\_куда\_пишем**

(>), (>>) - эти стрелки служат для **перенаправления.** Теоретически сейчас можно любой **output** перенаправить например в файл

Давай перенаправим Standard Error, вызовем любую команду с несуществующим значением, например: **cal qweqwe** , мы ожидаем Standard Error, но так как мы хотим чтобы вывод пришолся не по умолчанию в терминал, а в файл, наша команда будет: **cal qweqwe 2>> error.txt -** обязательным условием является наличие цифры **2** она даёт понять системе с каким outputam мы работаем, в частности сейчас с Standard Error, если цифра будет 1 то данный пример не сработает, так-как 1 это Standard Output, а мы работаем с Standard Error. **Важно** если записываешь ошибки в файл следить за наличием именно >> а не > так как при одном значке постоянно будет переписываться файл и не увидишь сколько на самом деле было ошибок.

— **перенап-ние outputa выполнения команды, если Standard Output и если Standard Error**—

**cal qweqwe 1>>FromUbuntu.txt 2>>error.txt -** Также перенаправление можно осуществлять комбинированно, в данном случае если команда будет выполняться штатно(без ошибок) то output пойдёт в предназначенную для него папку **FromUbuntu.txt**, если же возникнет ошибка то она выведется в **error.txt.**

**Примеры Перенаправления inputa:**

**Перенаправление inputa достигается указанием стрелочек << или <,** только для чего какая не понятно, использую <.

**Перенаправления inputa - это чтение не из консоли а из файла**

—---------—--------------------------------**выводим input из файла в терминал**—---------------------------------------

**cat 0< error.txt** - команда cat по умолчанию принимает какой-то ввод (input) с консоли, но c помощью **0<**  мы сообщаем что надо читать ввод не с консоли а из файла(Standard input 0), далее < и название файла откуда считывать, далее инф. из файла отобразится в терминале.

—------------------------------------------**перенаправление из файла в другой файл**—--------------------------------

Можно взять input из какого-то файла и вывести его не в консоль, а в какой то другой файл. Крутяк правда))) пример:

**cat 0< error.txt 1>> NewFile.txt** - здесь **cat** возьмёт за **input** те данные которые лежат в error.txt и выводит (сделает output) их в NewFile.txt, Почему **1>>**? Наверное потому что 1-отвечает за Standard output, а из файла читается этот самый standard output, если поставить 2 то не работает, так как 2 относится наверно к ошибкам при выполнении команд на терминале.

—-----------------------------

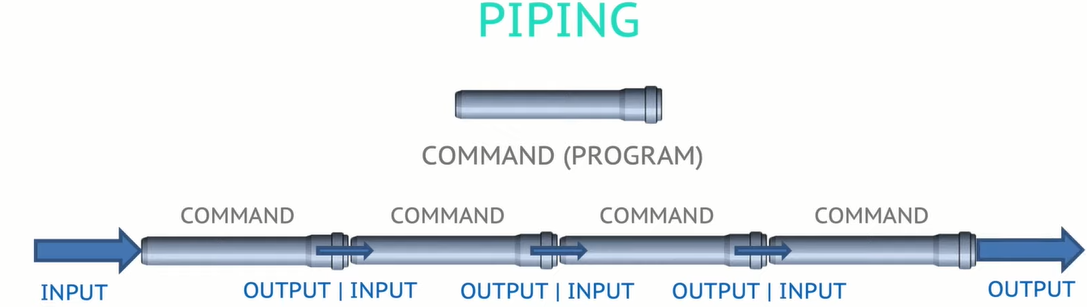
**tty** - выводит (Standard outputam -т.е. в терминал) информацию где расположен данный терминал.

-—----------------------------**перенаправление чего-то в другой терминал**—-------------------------------------------

Также перенаправлять input (c консоли или терминала)можно не только в файл но и в другой терминал. Например

**cat 0< название\_файла\_откуда\_читать 1> месторасположения\_терминала** - перенаправляем содержимое файла в какой то терминал (расположение терминала можно узнать по команде tty), Почему **0<** - так-как работаем с standard input, а не с предопределёнными значениями к командам. Почему 1>, наверно это Standard output и наверно потому что мы выводим в терминал, по идее по умолчанию выводится в терминал, возможно 1 можно опустить (не указывать), так как мы и так хотим вывести в терминал, проверю, да можно опустить. Но можно и заменить на 2, в этом случае будет выводится в терминал в котором прописывается команда, а не в тот куда отправляем, наверно из-за того что 2 это всётаки Standard Error. ХЗ

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

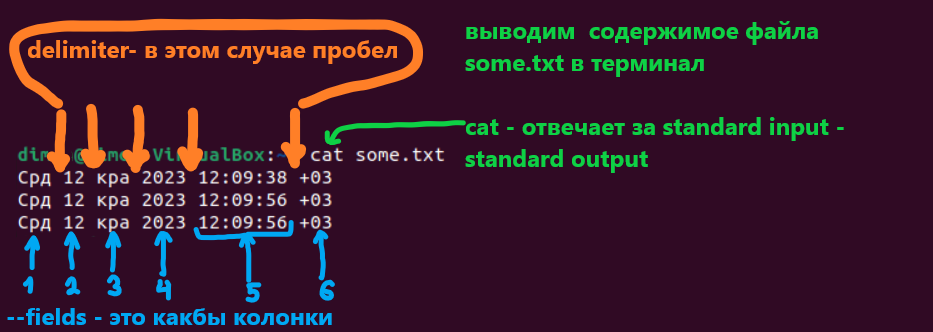


Представим что одна труба это одна команда, в которой есть входные данные (input) и выходные данные (output), соответственно output от одной команды будет (может быть) inut для другой. Таким образом мы как бы строим по очередности выполнения команд (как трубопровод) на выходе которого мы также получим какой то output, поэтому и называется piping.

**cut** - Вывести **выбранные** части строк из указанного ФАЙЛА в стандартный вывод. т.е. выводит какуюто инф в заданном порядке в терминал.

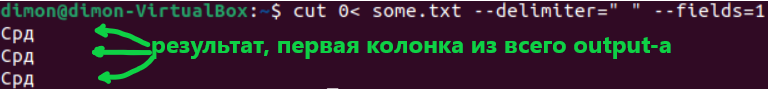
**date 1>> название\_файла** - запишим output команды date в файл, повторим 3 раза

**cat 0< название\_файла** - выведем всю имеющуюся инф. в терминал



Теперь я хочу чтоб данные из файла были выведены не целиком а только дни недели, для этого у cut есть опция –delimiter в значении указывается тот делимитер (имеется в виду символ/символы) по которому будет разделятся весь output. Здесь делиметром выступает пробел, смотри картинку. С помощью делиметра весь **output** делится на колонки с условным порядковым номером начиная с 1, данная опция называется **fields**.

**cut 0< some.txt –delimiter=” “ –fields=1**



**Т**еперь попробуем всё это же но с помощью **piping-a**

**date | cut –delimiter=” “ –fields=1** -единственное отличие теперь является то, что мы не считываем ничего из файла. cut - вывело свой output в терминал значок | обозначает что этот output был передан на вход команде cut как input, команда cut хоть и имеет настройки но по итогу отработав, тоже вывело свой output в терминал.

**date | cut –delimiter=” “ –fields=1 1>>название\_файла\_куда\_хотим\_сохронить**  - как вариант предыдущей команды, полученный output можно положить в папку, отличие состоит только в редиректе в конце, можно редирект делать и в другом месте, но не стоит так как **после редиректа пайпинг не работает далее приведу пример**

**date 1>> Data.txt | cut --delimiter=" " --fields=1,3** ← в этом случае пайпинг работать не будет, так как перед пайпингам стоит редирект, выполнится команда редиректа, т.е. до **|** значка пайпинга. Но и такого поведения можно добится с помощью команды tee

**date | tee Datka.txt | cut --delimiter=" " --fields=1,2** - до конца не понятно, похоже,что **| tee** является частью пайпинга но при этом заменяет символы **1>,** что обозначает создание и запись в файл.

**rm название\_файла** - удаляет файл, бля на линукс всё работает чётче. Весь интерес в том что **название\_файла**

является аргументом/**операндом,** а не input-ом, от какой то команды и соответственно не может быть использована в пайпинге, так как там используется input выполнения какой-то команды.

**cat название\_файла** - выводит на экран содержимое файла, создадим файл и запишем от руки в него название другого имеющегося файла который планируем передать как инпут. Т.е. я планирую вызвать команду которая прочитает содержимое файла где записано название другого файла.

**cat Dajta.txt | rm** - dajta.txt записано название файла которое я хочу с помощью пайпинга передать как input для rm и чтобы rm его удалил. но получаю ошибку, так-как оказывается есть input разный что ли, т.е. для rm надо аргументы, а у нас приходят эти данные не как аргументы. Здесь приходит на выручку команда xargs, которая ставится сразу после знака пайпинг

**cat Dajta.txt | xargs rm** - вот- так удаление пройдёт успешно, чтобы данные при пайпинге перешли в какую то команду которой необходимы аргументы в виде аргументов, то надо использовать команду **xargs.** Иначе возможна ошибка rm: missing operand

Всем привет! Смотрю канал в 2023, Сергею спасибо за работу+лайкос

1- Щас можно интэгрировать UBUNTU в винду →<https://www.groovypost.com/howto/install-and-start-bash-in-windows-10-anniversary-update/#:~:text=Start%20Bash%20in%20Windows%2010&text=Click%20Start%2C%20All%20Apps%2C%20under,type%3A%20bash%20then%20hit%20Enter>.

2- cal так и не заработал, там надо обновить репозиторий, не сложно → <https://itsfoss.com/unable-to-locate-package-error-ubuntu/>

3 - @DIMIDROL\_J <- мой телеграм, все кто учит автомэйшн тэстирование айда дружить, покажу/раскажу/подскажу чего понимаю, спрошу чего сам не догнал, вместе продуктивней. Всем добра и бабла.