Сегодня мы поговорим про перенаправление, запись и ДОЗАПИСЬ файл (тут важно усвоить разницу между этими двумя командами), поговорим про внимательность к символам > и >>.
Для того, чтобы понять то, о чём мы будем тут говорить, важно знать, откуда берутся данные, которые можно перенаправлять, и куда они идут. В Linux существует три стандартных потока ввода/вывода данных.

Первый — это стандартный поток ввода (standard input). В системе это — поток №0 (так как в компьютерах счёт обычно начинается с нуля). Номера потоков ещё называют дескрипторами. Этот поток представляет собой некую информацию, передаваемую в терминал, в частности — инструкции, переданные в оболочку для выполнения. Обычно данные в этот поток попадают в ходе ввода их пользователем с клавиатуры.

Второй поток — это стандартный поток вывода (standard output), ему присвоен номер 1. Это поток данных, которые оболочка выводит после выполнения каких-то действий. Обычно эти данные попадают в то же окно терминала, где была введена команда, вызвавшая их появление.

И, наконец, третий поток — это стандартный поток ошибок (standard error), он имеет дескриптор 2. Этот поток похож на стандартный поток вывода, так как обычно то, что в него попадает, оказывается на экране терминала. Однако, он, по своей сути, отличается от стандартного вывода, как результат, этими потоками, при желании, можно управлять раздельно. Это полезно, например, в следующей ситуации. Есть команда, которая обрабатывает большой объём данных, выполняя сложную и подверженную ошибкам операцию. Нужно, чтобы полезные данные, которые генерирует эта команда, не смешивались с сообщениями об ошибках. Реализуется это благодаря раздельному перенаправлению потоков вывода и ошибок.

Как вы, вероятно, уже догадались, перенаправление ввода/вывода означает работу с вышеописанными потоками и перенаправление данных туда, куда нужно программисту. Делается это с использованием символов `>` и `<` в различных комбинациях, применение которых зависит от того, куда, в итоге, должны попасть перенаправляемые данные.

важно заметить, что перенаправление > используется гораздо чаще (чем < - это вычитка из файла) и о нем мы поговорим чуть подробнее.

Piping and redirection

Что важно знать про pipe?

Пайпинг знак |

при использования pipe результат выполнения одной команды подается на вход другой.

Это и есть перенаправление результата команды для выполнения другой. Т.е. - это не запись в файл, это возможность продолжить работу с данными, которые мы получили при помощи предыдущей команды.

Теперь поговорим про перенаправление

Команда:

Date

как она работает? Мы же не запускаем каких-то дополнительных программ?

Все просто. Ты отправляешь текст на сервер. На нем есть оболочка, в которой выполняются команды. Эти оболочки есть разные. Оболочка sh/ ash/ bash. Все зависит от дистрибутивов.

ОС получает команду, что нужно выполнить date. Файла исполнения нету, но благодаря этим встроенным оболочкам ОС сама понимает, что и откуда нужно запускать. Магия! И облегчение работы для программиста.

Перенаправление.

Теперь давайте разберемся, что такое перенаправление.

> - записать в файл. Мы можем записать в файл и то, что видим в файле и результат выполнения команды.

ПРИМЕР:

Создадим файл и запишем в него (не ДОзапишем, а именно запишем) результат выполнения команды date.

```
localhost:~# date > /tmp/file
```

Дописать в файл.

```
localhost:~# echo hello world
hello world
localhost:~# echo "hello world"
hello world
localhost:~# echo "hello world" >> /tmp/file
localhost:~# cat /tmp/file
Sat Jan 29 18:23:43 UTC 2022
hello world
```

Двойное перенаправление.
Если записать в файл, и не использовать двойное перенаправление, то можно убить все данные в файле. Всегда следите, когда дозаписываете, иначе отпуск в теплом декабре вам обеспечен (=

>
Save output to a file.
>>
Append output to a file.

Теперь давайте узнаем, как сделать так, чтобы записать или дозаписать в файл определенные данные.
Иногда случается так, что нам сначала нужно найти какие-то данные (будь то слова, символы или строчки) и перенести их в файл для последующего разбора полетов.
И тут нам важно знать, как работает tail и head

Как работает tail and head

Cat /etc/group
эта команда выведет нам на экран все, что есть в этом файле.

Теперь посчитаем количество строк в этом файле
и тут мы воспользуемся pipe, который перенаправит результат вывода всех данных в файле на новую команду, которая посчитает количество всех строк.

cat /etc/group | wc -l
И результат мы передаем другой команде через pipe |
Где wc -l считает количество строк

cat /etc/group | wc

Покажет количество строк, слов и количество символов

Wc - word count количество строчек. И с ключом -l оно просто покажет количество строк

Вот так можно пользоваться pipe.

Теперь другое

cat /etc/group | head

Это выведет на экран первые 10 строк. У head назначение выводить первые 10 строк, но можно выводить любое количество.

Теперь можно выводить с помощью tail.
cat /etc/group | tail
tail так же по умолчанию показывает 10 строк

Возможно, вам может показаться, что этими командами можно выполнять вывод только строки в файле, но это не так.

Этими командами можно выводить даже историю и историю определенного количества программ в истории.

history | tail -3
Тут мы показали последние три команды из истории.
Вместо cat можно использовать любую команду, которая возвращает нам какой-либо результат.

Теперь мы допишем в файл эти три строки из истории, используя >>
History | tail -3 >> /tmp/file

Перенаправление можно использовать и в обратном направлении, но используют первый вариант
wc -l < /etc/group
Вышеназванный вариант используется крайне редко.

Теперь про grep

Grep - это функция поиска по символам. Используется при поиске. Например мы можем показать строки в /etc/group , в которых встречается слово root
И использовать мы будем тоже через pipe.

cat /etc/group | grep root

Когда мы делаем grep - показывается слишком много информации. Для вывода определенных слов, мы можем использовать флаг -w, который будет искать конкретное слова. Тоесть это будет не часть слова, а конкретное слово (слово целиком!!!)

У grep есть куча флагов, которые помогают нам в сортировке. Рассмотрим пока -w.

Grep -w
cat /etc/group | grep -w user2

Мы **можем сортироваться и фильтроваться огромное количество раз. Например, отфильтруем не только слово user2, но и в каком количестве строк оно встречается**

cat /etc/group | grep -w user2 | wc -l
А после мы может результат дописать в файл
cat /etc/group | grep -w user2 | wc -l >> /tmp/file

И проверим
Проверяем, конечно же cat /tmp/file

Забыл сказать на лекции:

У tail есть интересный ключ -f, который поможет отслеживать в реальном времени текст в файле.

-v исключить строчки из вывода.

Команда, показывающая насколько занят диск.
df
Можно использовать с ключом df -h, тогда он будет в понятном нам виде, человеческом.

Все привыкли работать в ОС и видеть, что результат выполнения команды вылазит на экран в виде графики.
В linux так не происходит. Ты отправляешь текст и результат возвращается текстом. И благодаря pipe ты можешь это почитать, отфильтровать и записать в файл.

Тут важно заметить, что в Linux большинство проблем из-за того, что заканчивается место.
Сейчас мы с помощью других команд и pipe отфильтруем результат

Можно заметить**, что это можно сделать через tail, но это долгое и не оптимальное решение. Бывают случаи, когда нужно сделать быстрее.**
df -h - в человеческом виде
df -h | head -2
Тут мы показываем две строчки текста
df -h | head -1
Тут мы показываем только первую строчку

Если мы добавим tail -1, мы получим нужный результат, но это будет не совсем правильный алгоритм (Потому что иногда бывает, что в файлах с

логами что-то постоянно меняется и строчки могут “убегать” и будут выведены не совсем те, которые нам нужно). Но в нашем случае, для эксперимента схема рабочая.
df -h | head -2 | tail -1

Тут мы показываем все строчки, в которых есть символ /
df -h | grep /

Почему лучше использовать grep? Вариант с ним отработает на любой системе потому что мы ищем чётко инфо о занятом месте в корневом разделе файловой системы.
При grep меняется логика фильтрации. Он показывает без подгона результатов сразу строку.

Отсортируем строчки, где будет только / и в этом нам поможет ключ -w (word)
df -h | grep -w /

Теперь отсечем все лишнее, чтобы оставить цифру 47

для этого нам поможет awk.

awk - это утилита/язык для извлечения данных.

Говоря проще awk помогает относиться к тексту, как к таблице
и сейчас мы отфильтруем так, чтобы он показал нам пятый столбец.

df -h | awk '{print $5}’
Нам показало столбец.

Но как показать 47%?

А вот так:
df -h | grep -w / | awk '{print $5}’
Теперь избавимся от знака %

Чуть позже вернемся, а пока поговорим о замене символов и слов

Вернемся к команде
cat /etc/group | grep -w user2
Теперь мы поменяем слово user2 на имя.
cat /etc/group | grep -w user2 | sed ‘s/user2/ivan/g'

Что это значит? Sed - stream editor
Мы отфильтровываем user2 в папке и файле /etc/group,а затем при помощи команды sed меняем имя user2 (s-swap-замена) на имя ivan, причем параметром g (он называется глобал), меняем все слова user2 на ivan.

Теперь мы уберем знак % из строчки с цифрой 47
Точнее не уберем, а заменим его ничем. Причем ничто мы введем без пробела!

```
df -h | grep -w / | awk '{print $5}' | sed 's/%//g'
```

и сделаем уже известные нам команды записи в файл
history > /tmp/history.txt
export_file /tmp/history.txt
теперь мы знаем, что это за знак больше.