**cd**

**mkdir -p**

**touch**

**cp**

**mv**

**rm**

**date**

**tree**

**ls**

**history**

* **- help**

**mkdir** (make directory) - создавать папки

**rmdir** (remove directory) - удалять папки

**touch** - создавать файлы. Как я уже сказал, любой файл создается в Linux, как текстовый и открывается текстовым редактором, так как в нету привязки к расширению (как в windows), где система сначала нюхает расширение и делается вывод, какой программой необходимо файл запускать.

**cp** (copy) - команда копирования

**mv** (move) - команда перемещения

**rm** (remove) - команда удаления файла. Иногда может помогать нам удалять целые каталоги, но для этого нужен флаг **-rf**

**Так как командная строка не имеет функции отмены - всегда нужно понимать, что и зачем ты делаешь. Ведь “вернуть” удаленный файл или папку не будет возможности.**

Открываем наш тестовый терминал по ссылке:

[**https://bellard.org/jslinux/index.html**](https://bellard.org/jslinux/index.html)

**Заходим в папку home, как мы помним - командой cd**

140 cd /home

**создаем папку user1**

141 mkdir user1

**смотрим, что папка создана, и что еще есть у нас в этой папке**

142 ls

**переходим в папку user1**

143 cd user1

**создаем еще одну папку task1**

144 mkdir task1

**заходим в нее**

145 cd task1

**и уже в этой папке создаем файл (в данном случае, для наглядности, с расширением \*.txt)**

146 touch file1.txt

*Давайте посмотрим, где мы находимся. Какая команда нам в этом поможет?* **pwd**

147 pwd

**так же давайте проверим наши разрешения и посмотрим на размер созданного файла. Он будет нулевым, так как в нем пусто.**

148 ls -l

Флаги и аргументы. Флаги или ключи идут с минусом -ключ меняет визуальное отображение. К примеру команда ls -l (l - long) - подробная информация. Есть еще **-a** где показываются все файлы, в том числе скрытые. Есть псевдоним ls -la

149 ls -a

151 ls -l

Если случилось страшное и наш экран забит кучей лишней информации, то всегда можно сделать команду clear и наш терминал станет девственно чистым.

Ctrl l - очистить экран и **clear**

Так же есть еще одна интересная фишка - это ветвления папки:

Помогает нам в этом команда **tree**:

**Делать мы этого не будем в корневом каталоге, так как у нас терминал начнет показывать абсолютно все файлы!**

Если использовать эту команду со слешем /, то она показывает ветвления от папки, которую ты запросишь в терминале.

**пример: tree /home/**

Важный момент! Tree есть не во всех дистрибутивах. Пробовать можно во всех, но иногда результата мы не увидим. Пугаться не стоит. Просто мы используем такую версию linux, где tree нету.

Должно получиться в папке home несколько папок. Каждый файл создается чуть быстрее.

В терминале можно посмотреть дату и время. Это, если ты вдруг забыл, сколько сейчас времени и когда там уже перерыв. Делается это командой **date**:

152 date

Так же можно посмотреть, что мы там понавводили за все занятие и в каком порядке. Для этого нам поможет команда **History**:

155 history

**Теперь давайте попробуем разобраться, как же создавать файлы и папки одной строкой, не заходя каждый раз в новосозданную папку и подпапки.**

156 mkdir -p /home/user2/task2/

157 touch /home/user2/task2/file2.txt

158 mkdir -p /home/user3/task3/

флаг **-p** создать путь сокращение от path если нужно создать папку и путь

Если просто папку mkdir то создается просто папка без вложенности.

иногда используется для неотображения ошибок в некоторых версиях Linux

Второе назначение - убедиться в наличии пути

-p используют для скриптов, чтобы не возвращать ошибку, а убеждаться в наличии путей.

**Напомню, что такое скрипт:**

Скрипт - это небольшая программа, которая содержит последовательность действий, созданных для автоматического выполнения задачи.

**создадим папку**

159 mkdir /home/user2/task2/

Увидим ошибку. Не пугаемся, так выглядит ответ, если файл или папка уже есть.

Тут важно сказать о путях. В линуксе существует Полный (он же абсолютный путь) и относительный.

* **Полный, абсолютный путь linux от корня файловой системы** - он начинается от корня "/" и описывает весь путь к файлу;
* **Относительный путь linux** - это путь к файлу относительно текущей папки, такие пути часто вызывают путаницу.

Простой пример. Сидите вы дома, и вдруг стук в дверь. Вы открываете, а там полиция. Вас кладут лицом в пол и спрашивают:

* Вася?

А вы им - НЕТ! Вася живет на два этажа выше во второй квартире слева! - это относительный путь.

Но если вы скажете - Вася живет на пятом этаже во второй квартире слева! - это уже будет полный.

* **Путь относительно домашний папки текущего пользователя.** - путь в файловой системе, только не от корня, а от папки текущего пользователя.