УРОК 4. ТЕКСТОВЫЕ РЕДАКТОРЫ НА ПРИМЕРЕ VI И NANO

ТЕКСТОВЫЙ РЕДАКТОР VI

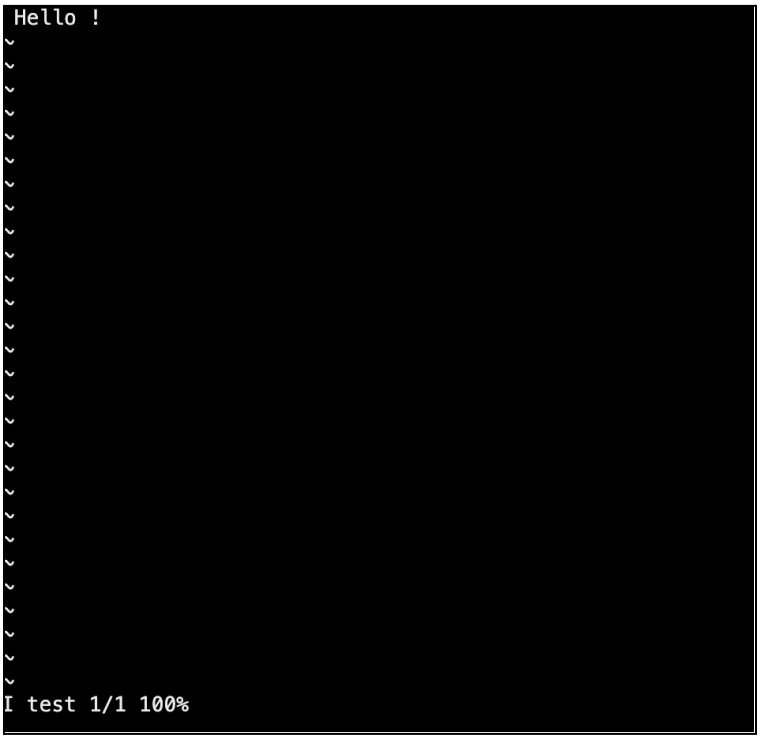
В Linux существует множество текстовых редакторов, каждый из которых предназначен для определенных задач и соответствует определенным предпочтениям пользователей.

Файлы в ОС Linux являются текстовыми.

vi (и его усовершенствованный клон vim) является одним из наиболее популярных текстовых редакторов в мире UNIX-подобных операционных систем, включая Linux.

Редактор vi – один из старейших **Initial release** 1976; 48 years ago

Именно так выглядит окно текстового редактора vi :



Vim: Vim (Vi Improved) является усовершенствованным клоном классического редактора vi. Vim предоставляет мощные возможности редактирования, включая множество команд, подсветку синтаксиса и расширяемость через плагины, многоуровневый процесс отмены ранее выполненных действий (undo),

использование нескольких окон редактирования и многое другое.

Режимы работы:

- Командный режим (Command Mode): В этом режиме клавиши используются для выполнения команд, таких как перемещение по тексту, удаление символов, копирование и вставка.
- Режим вставки (Insert Mode): Здесь вы можете вводить и редактировать текст, как в обычном текстовом редакторе. Для перехода в этот режим нажмите клавишу "i" в командном режиме.

Основные полезные команды для работы в vi :

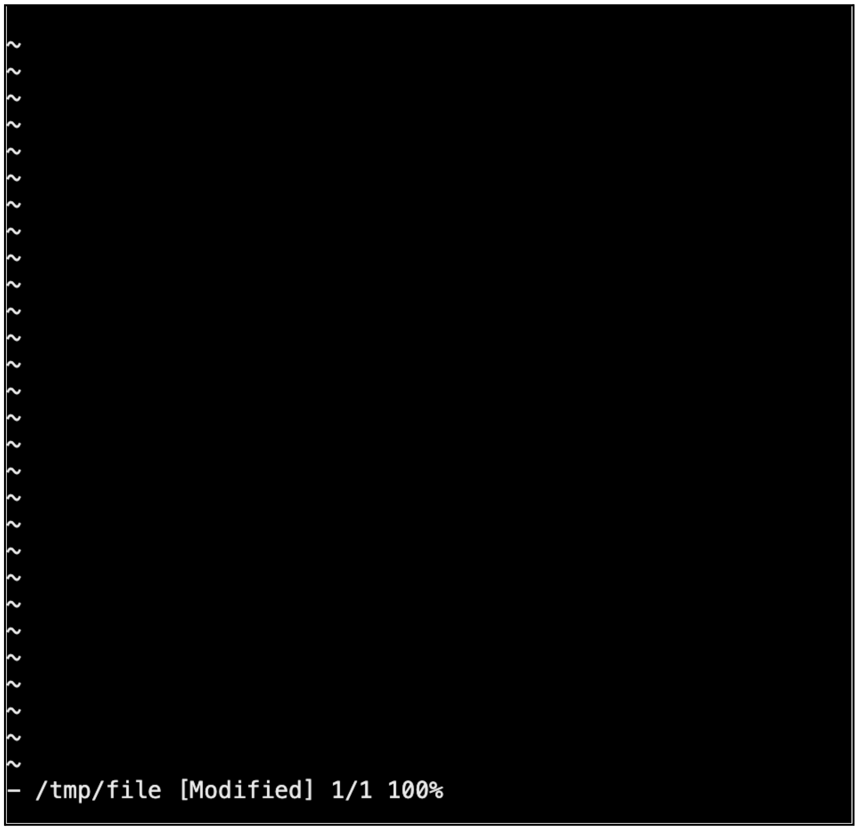
- ZZ (Note: БОЛЬШИЕ БУКВЫ) - Сохранить файл и выйти из редактора
- :q!: Закрыть файл, отменить все изменения с момента последнего сохранения.
- :w: Сохранить файл, но не выходить из редактора.
- :wq: Сохранить файл и выйти из редактора.
- :x: То же самое, что и :wq - сохранить файл и выйти из редактора.
- G (Shift + G): Перейти к последней строке в файле.
- /: Начать поиск в тексте (после ввода /, введите текст для поиска, затем нажмите Enter; используйте n для перехода к следующему совпадению).
- i: Войти в режим вставки (режим редактирования), чтобы начать вводить текст. Нажмите Esc, чтобы выйти из режима вставки.
- u: Отменить последнее действие (отменить изменение).
- dd: Удалить текущую строку. (i mode not needed)
- yy: Копировать текущую строку.
- p: Вставить содержимое буфера после текущей строки.
- Del - удалить один знак (i режим (вставки) не требуется)

vim является мощным и гибким редактором, но из-за особенностей командного режима он может потребовать времени для освоения.

Попробуем поработать с vi , используя основные команды:

vi /tmp/file

Тем самым мы создадим текстовый файл file в папке /tmp и откроем его для редактирования. Если мы не сохраним его и выйдем из редактора, то и файла не останется.



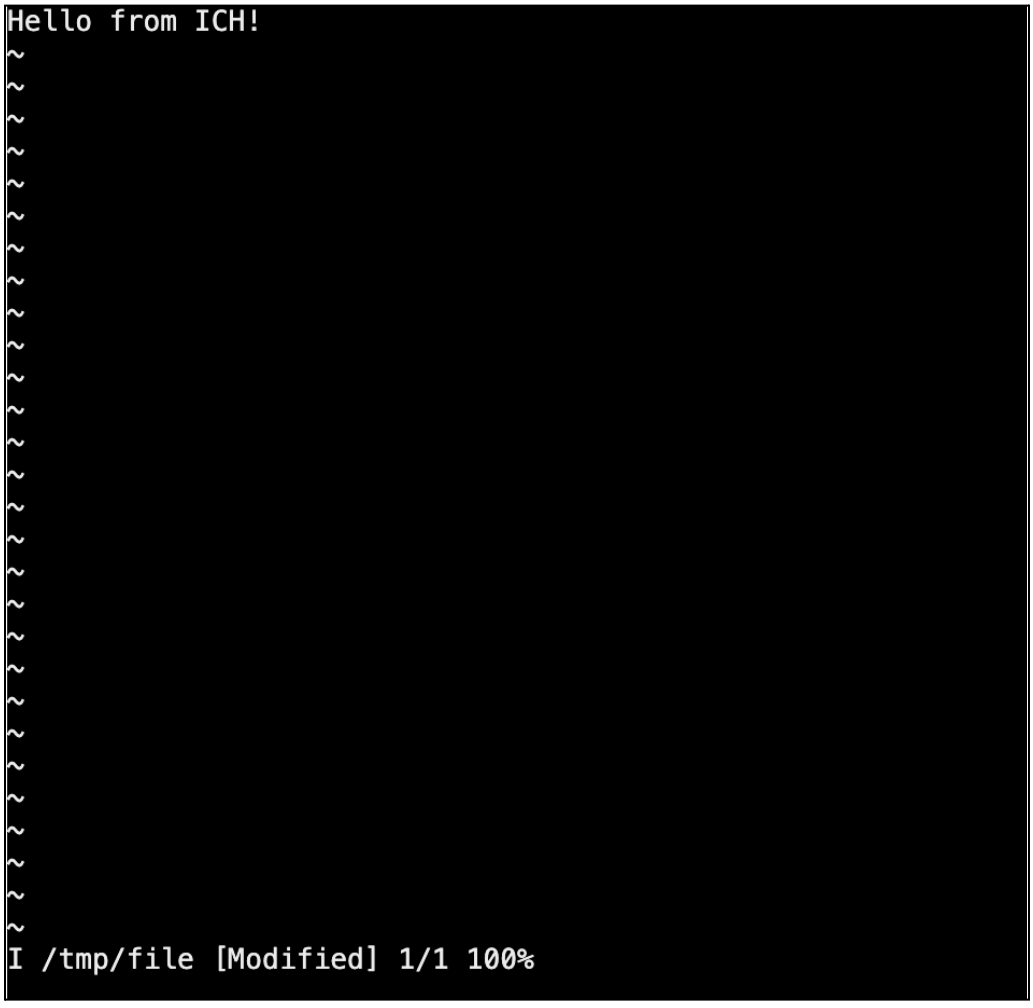
Пустой экран редактора, он занимает всю площадь нашего экрана.
Тильды видны только нам и указывают на то, что это пустые строки. На самом деле тильд нету и если мы вдруг захотим распечатать что-то, то там будет пусто.
Внизу имя и путь к файлу и что курсор находится на первой строке из всех строк и это 100% от всего документа. То есть документ новый и чистый.
В левом нижнем углу символ, который как и кавычки в Linux, можно спутать с другим. Это не тильда, это минус. Он нам говорит, что мы находимся в командном режиме. Еще некоторые его называют режимом просмотра. Когда мы видим такой символ, это означает, что Vi ждет от нас команд, либо мы просто зашли сюда посмотреть на документ.

Режимы просмотра и редактирования (командный и вставки):
Перейти в другой режим мы можем клавишей i

Находясь в режиме ввода и редактирования можно написать какой-то текст или редактировать существующий.

Напишем Hello from ICH!

Обратите внимание на индикацию режима работы внизу слева - вместо “-” там теперь I, что означает, что мы в режиме вставки.



Выходим из режима редактирования, сохраняем документ и выходим из текстового редактора.

Первое - нам нужно выйти из режима редактирования. В левом нижнем углу буква I. Теперь нажмем на esc. И видим “-” теперь мы в командном режиме.

Теперь выходим из редактора с сохранением. Для этого используем один из многих способов выйти:

- ZZ (Note: БОЛЬШИЕ БУКВЫ) - Сохранить файл и выйти из редактора
- :w: Сохранить файл, но не выходить из редактора.
- :wq: Сохранить файл и выйти из редактора.
- :x: То же самое, что и :wq - сохранить файл и выйти из редактора.
- :q!: Закрыть файл, но отменить все изменения с момента последнего сохранения.

Мы попадаем в терминал и можем вводить другие команды:

Добавим дату в этот документ и посмотрим на результат:

date >> /tmp/file

vi /tmp/file

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

```
Hello from ICH!
Tue Jan 30 15:57:24 UTC 2024
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
- /tmp/file 1/2 50%
```

В открывшемся документе изменим содержимое (заменив ICH на ваше имя) и удалим строку с датой при помощи dd.

Откатить изменение можно при помощи u в командном режиме.

Сохраняем и выходим.

ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

Попробуйте самостоятельно:

1. Создать текстовый файл file в папке /tmp и открыть его для редактирования.
2. Перейти в другой режим.
3. Написать: Hello from ICH!
4. Выйти из режима редактирования
5. Сохранить документ и выйти из текстового редактора
6. Добавить дату в этот документ и посмотреть на результат
7. В открывшемся документе изменить содержимое (заменить ICH на ваше имя) и удалить строку с датой при помощи dd.
8. Сохранить и выйти

ТЕКСТОВЫЙ РЕДАКТОР NANO

Nano - это простой текстовый редактор в командной строке, который предоставляет базовый, легкий в использовании интерфейс. В отличие от более мощных редакторов, таких как Vim, Nano не требует изучения сложных комбинаций клавиш, что делает его отличным выбором для новичков.

Единственным минусом, который можно назвать, является его отсутствие на некоторых дистрибутивах.

Несмотря на очень маленький размер - около 600 килобайт, некоторые разработчики, делая дистрибутив экономят даже на этом. И не включают этот редактор в стандартный пакет.

Запуск nano:

nano /tmp/file

Интерфейс nano:

- Сверху название файла и версия редактора.
- После входа попадаем сразу в режим редактирования и можем редактировать документ.
- Внизу есть подсказки, например, как выйти из редактора
- Чтобы выйти нужно нажать ctrl+x. Это и есть ^X Exit

ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

1. При помощи vi и nano создать текстовый документ: /home/text_editors
2. Добавить в него следующий текст при помощи vi и nano соответственно:

   I love vi
   и
   I love nano

PIPING: КОМАНДЫ DF, GREP, AWK, SED

Выведем информацию о занятом дисковом пространстве:

```
localhost:~# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                 4.9G      2.3G      2.6G  47% /
devtmpfs                 91.3M         0     91.3M   0% /dev
tmpfs                    91.4M      8.0K     91.4M   0% /run
none                     91.4M         0     91.4M   0% /dev/shm
localhost:~#
```

Задача - получить значение занятого дискового пространства для корневого раздела без знака % , а именно 47 и записать его в отдельный файл.

Для этого мы можем использовать следующую цепочку преобразований:

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

Разберем каждый из этапов в пайпинге подробней.

Команда df (от англ. "disk free") используется для отображения информации о доступном дисковом пространстве на файловых системах.
Наиболее частое использование - df -h
df -h Выведет информацию о доступном дисковом пространстве в человекочитаемом формате, а именно переводя килобайты в мегабайты, мегабайты, гигабайты и т.д.
Пример:

```
localhost:~# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                 4.9G      2.3G      2.6G  47% /
devtmpfs                 91.3M         0     91.3M   0% /dev
tmpfs                    91.4M      8.0K     91.4M   0% /run
none                     91.4M         0     91.4M   0% /dev/shm
localhost:~# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/root              5120000   2417708   2702292  47% /
devtmpfs                 93464         0     93464   0% /dev
tmpfs                    93620         8     93612   0% /run
none                     93620         0     93620   0% /dev/shm
```

Команда grep используется для поиска текстовых данных в файлах или выводе других команд.

Поиск строки в файле: `grep "pattern" filename.txt`
Поиск в выводе другой команды: `ls -l | grep "file"`

Grep имеет множество ключей, что предоставляет множество опций для точной настройки поиска.
Вот некоторые из них :

- -i или --ignore-case: Игнорирует регистр символов при поиске.
- -v или --invert-match: Выводит строки, не содержащие совпадений.
- -w или --word-regexp: Ищет только слова, а не части слов.
- -r или -R или --recursive: Производит поиск рекурсивно в поддиректориях.
- -A или --after-context=N: Выводит N строк после совпадения.
- -B или --before-context=N: Выводит N строк перед совпадением.
- -C или --context=N: Выводит N строк вокруг совпадения.

Возвращаясь к нашему примеру мы можем использовать grep для быстрой фильтрации и оставить лишь строку с корневым разделом.
Обратите внимание на использование ключа -w , иначе мы получаем все строки с символом /

```
localhost:~# df -h | grep /
/dev/root                4.9G      2.3G      2.6G  47% /
devtmpfs                91.3M         0     91.3M   0% /dev
tmpfs                   91.4M      8.0K     91.4M   0% /run
none                    91.4M         0     91.4M   0% /dev/shm
localhost:~# df -h | grep -w /
/dev/root                4.9G      2.3G      2.6G  47% /
```

awk - это утилита командной строки и язык программирования, который используется для обработки и анализа текстовых данных в текстовых файлах и выводе команд. Название awk является акронимом, образованным из первых букв фамилий его создателей.

awk может считывать строки текста, разбивать их на поля (колонки) и выполнять различные операции над этими данными.

Возвращаясь к нашему примеру мы представляем строку как таблицу с одной строкой и 6 колонками с пробелами-разделителями . Нам нужна 5 колонка:

```
/dev/root                 4.9G      2.3G      2.6G  47% /
localhost:~# df -h | grep -w / | awk {'print $5'}
47%
```

sed (Stream Editor) - это утилита предназначенная для выполнения текстовых преобразований в потоках данных. sed работает по принципу чтения входных данных построчно.
sed может использоваться для поиска и замены текста в строках. Это может быть как простая подстановка, так и удаление и вставка текста.

По умолчанию результат работы выводится в стандартный вывод.

Пример: замена слова "apple" на "orange" из файла filename.txt
sed 's/apple/orange/g' filename.txt

Синтаксис sed 's/apple/orange/g' -
s - swap - подмена
/apple/orange/ - между знаками / мы размещаем “что” заменить и “на что” заменить.
/g - g после последнего слеша (/g) означает "глобальную" замену. Это означает, что sed будет заменять все вхождения шаблона (в данном случае, "apple") на указанную строку (в данном случае, "orange"), а не только первое вхождение в каждой строке.

Или еще один пример , мы хотим заменить user2 на слово Ivan
cat /etc/group | grep -w user2 | sed ‘s/user2/ivan/g'

```
localhost:~# cat /etc/group | grep -w user2 | sed 's/user2/Ivan/g'
Ivan:*:1000:
Ivan:*:1000:
Ivan:*:1000:
```

Если необходимо перезаписать исходный файл результатом работы команды sed , то необходимо добавить ключ -i , но необходимо передать файл, над которым мы проводим эти операции:

```
sed -i 's/user2/Ivan/g' /etc/group
```

```
localhost:~# sed -i 's/user2/Ivan/g' /etc/group
localhost:~# cat /etc/group | grep -w Ivan
Ivan:*:1000:
Ivan:*:1000:
Ivan:*:1000:
```

Итого : вывод df -h был отсортирован и отфильтрован до одного конкретного значения

Из:

```
localhost:~# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                 4.9G      2.3G      2.6G  47% /
devtmpfs                 91.3M         0     91.3M   0% /dev
tmpfs                    91.4M      4.0K     91.4M   0% /run
none                     91.4M         0     91.4M   0% /dev/shm
```

Мы получили:

```
localhost:~# df -h | grep -w / | awk {'print $5'} | sed 's/%//'
47
```

ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

Найдите **НЕ**верное утверждение:

Ответ напишите в чат.

1. Поток № 2 - это поток данных, которые оболочка выводит после выполнения каких-то действий.
2. Grep - это функция поиска по символам.
3. awk - это простая, но мощная утилита, которая анализирует текст и плавно преобразует его.
4. df - команда, показывающая насколько занят диск.

ПРАКТИЧЕСКАЯ РАБОТА

1. Создайте файл с названием myfile.txt по пути /root/test3/
2. Запишите в этот файл вывод команды df -h
3. Допишите в файл 9 первых строк из списка всех объектов корня
4. Допишите в файл количество символов из трех последних строк файла /etc/group