

Улучшенный визуальный редактор (vim)

Исходные тексты программ являются обычными текстовыми файлами. Для их создания и редактирования используются текстовые редакторы. В дистрибутивы операционной системы Линукс включаются различные графические текстовые редакторы, например, `medit`, `gedit`, `leafpad`, `mousepad`. Для разработки скриптов и программ часто используются текстовые редакторы встроенные в интегрированные среды разработки, такие как `KDevelop`, `QTCreator`, `Eclipse`, `Geany`.

Для работы в режиме терминала наиболее популярными текстовыми редакторами является `emacs`, встроенный редактор `Midnight commander` и `nano`. Однако первенство по популярности остаётся за `vi`, и, примерно с 1991 года, за его улучшенной версией `vim`.

`Vim` — это сокращение от `visual improved` (визуальный улучшенный). `Vim` является наследником `vi`, который поставляется со всеми версиями POSIX систем, начиная с 1976 года. Современные дистрибутивы Линукс часто содержат `vim` вместо `vi`.

Редакторы `vi` и `vim` имеют одинаковые комбинации «горячих клавиш» и логику работы, но `vim` более функционален и имеет ряд нужных возможностей, отсутствующих в `vi`. Целесообразно сначала проверять наличие в системе именно `vim` и только в случае его отсутствия использовать `vi`. Далее будет рассматриваться редактор `vim` без дополнительных плагинов.

`Vim` готов к использованию сразу после установки, но повысить удобство его применения можно с помощью файла конфигурации, который индивидуален для каждого пользователя. Файл конфигурации расположен в каталоге пользователя и называется `.vimrc`. Точка в первом символе названия файла означает что он скрытый и не отображается при просмотре содержимого каталога без явного требования отображения скрытых объектов.

Автором используется файл `.vimrc` следующего содержания

```
1 set tabstop=8
2 set softtabstop=8
3 set wrap
4 set showmatch
5 set autoread
6 set title
7 set number
8 set backupdir=~/.vimbackup
```

Далее описано назначение параметров указанных в файле конфигурации. Каждая строка начинается со слова **set**, что означает установить параметр.

Строка **set tabstop=8** определяет количество пробелов, которыми символ табуляции отображается в тексте. Данный параметр особенно удобен, в случае если отступы в исходных текстах программ и скриптах выполняются с помощью символа **Tab**.

Строка **set softtabstop=8** количество пробелов, которыми символ табуляции отображается при добавлении. Данный параметр позволяет избежать появления смешанных отступов из пробелов и символов табуляции, которые возникают, например, при трёхкратном нажатии клавиши **Tab**.

Строка **set wrap** позволяет выполнять перенос строки, если она не вмещается в окно редактора по ширине. Выбор данного параметра определяется вкусом пользователя. При переносе строки номер строки на которую перенесён текст не проставляется.

Строка **set showmatch** включает подсветку для всех совпадений с шаблоном в процессе поиска в тексте некоторого фрагмента.

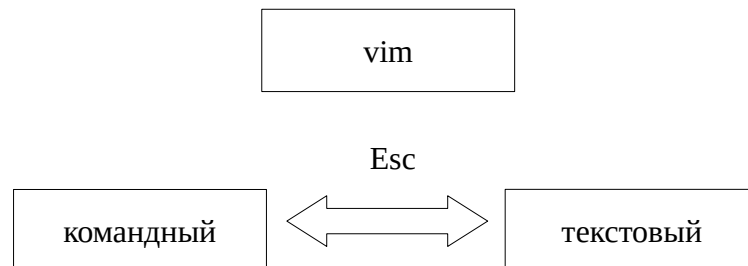
Строка **set autoread** предписывает редактору vim автоматически перечитывать файл каждый раз, когда он был изменён другой программой.

Строка **set title** отображает имя файла в заголовке файла.

Строка **set number** отображает номера строк файла открытого на редактирование.

Строка **set backupdir=~/.vimbackup** задаёт каталог в который сохраняются все временные файлы. Внимание! До использования этого параметра, необходимо создать каталог **~/.vimbackup**.

После создания и редактирования конфигурационного файла можно переходить к использованию vim. Программа vim имеет два основным режима работы: командный и текстовый. Переключение между ними производится с помощью клавиши **Esc**.

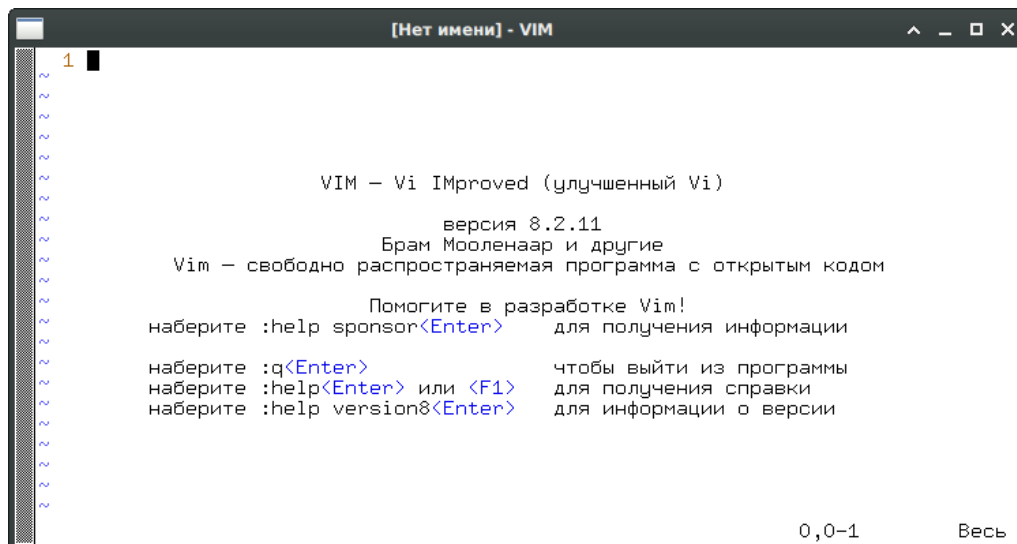


В командном режиме нажатие клавиш на клавиатуре приводит к изменению состояния программы и исполнению команд, соответствующих нажатым клавишам. В текстовом режиме производится перемещение по тексту, вставка, редактирование или удаление символов. Редактор имеют большое количество «горячих клавиш», которые выполняют действия, редко доступные в графических текстовых редакторах: удаление строки, переход на начало или конец абзаца и др. Для переключения между режимами используется клавиша **Esc**. Со времён использования vim на вычислительных машинах, оснащённых системным динамиком, существует шутка о том, что vim работает в двух режимах: в одном пищит, в другом портит текст.

Основная сложность при работе с vim следует из его способа взаимодействия с пользователей и состоит в том, что необходимо приучить себя всегда переводить программу в командный режим. В таком случае не придётся запоминать в каком режиме находится программа.

Далее описан запуск редактора vim и завершение сеанса работы с ним.

Для запуска редактора необходимо выполнить команду **vim** . После запуска выводится сообщение и текстовый редактор переводится в режим ввода команд.



Для завершения работы с программой без сохранения сделанных изменений необходимо выполнить следующие действия:

1. Ввести символ «:». Символ будет отображён в левом нижнем углу окна программы.

2. Ввести символы «q!». Они также будут отображены в левом нижнем углу программы.

В результате выполнения пунктов 1 и 2, в нижнем левом углу должна отображаться последовательность из трёх символов «:q!».

3. Нажать **Enter**.

После этих действий работа с редактором будет завершена.

В командах vim часто используются специальные символы, поэтому последовательности команд будут заключены в типографские кавычки, т. е. нажатие клавиши **q** будет обозначаться как «q».

В командном режиме необходимо отслеживать включённую раскладку клавиатуры. При наличии в командах символов, отличных от английских, команды выполняться не будут.

Программа vim использует терминологию, несколько отличную от современной общепринятой. Приведём главные понятия используемые vim:

- **буфер** – временное хранилище текста, копия отображаемого или редактируемого файла. Каждый редактируемый файл связан с единственным

буфером, но каждый буфер может отображаться в неограниченном количестве окон;

- **окно** – область экрана для просмотра и отображения одного буфера;

- **вкладки (сленг. табы)** – механизм группировки и переключения между группами окон;

- **регистр по умолчанию** – наименованное хранилище текста для множественных вставок, наподобие «буфера обмена» в офисных пакетах;

- **именованный регистр** – именнованное хранилище текста для множественных вставок. Также напоминает один из режимов работы «буфера обмена» в офисных пакетах;

- **аббревиатуры** – сокращения, которые при наборе отдельных слов путём замены разворачиваются в соответствующий текст;

- **изменение** – включает в себя вставку текста из регистра, аналог команды «вставка» в графическом текстовом редакторе.

Для получения исчерпывающей справочной информации необходимо после запуска программы vim ввести команду «**:help**». Для получения справки по конкретной команде «**:help W**». Данная последовательность выведет справку по команде «**W**». Для выхода из режима справки необходимо использовать комбинацию «**:q**».

Параметры задаваемые в файле .vimrc могут устанавливаться во время работы, однако их значения будут сохраняться только в текущем сеансе. После выхода из vim, все внесённые изменения будут забыты.

Для демонстрации изменения параметров vim необходимо выполнить следующие действия:

1. Запустить программу командой **vim** .
2. Нажать клавишу при включённой английской раскладке «**a**» .
3. Ввести две строки текста:

Первая строка.

Вторая строка.

Если файл `.vimrc` был настроен верно и действия под цифрами 1, 2, 3 выполнены правильно, должно получиться следующее.



Надпись внизу страницы -- ВСТАВКА -- говорит о том, что программа находится в режиме вставки, можно вводить текст.

Далее необходимо выполнить следующие действия:

4. Переключится на английскую раскладку клавиатуры;
5. Нажать клавишу **Esc** для переключения в режим ввода команд.
6. Ввести символы «**:set nonumber**».



7. Нажать клавишу **Enter**. После этого нумерация строк исчезнет, однако последняя введенная команда сохранится в нижней левой части окна программы.



В описанном выше примере используется команда **nonumber**, которая является обратной команде **number**, использованной в файле `.vimrc`.

Для выхода из программы необходимо перевести программу в командный режим клавишей **Esc**, ввести команду «:q!» и нажать клавишу **Enter**. После выхода из программы настройка `nonumber` будет сброшена. Команду завершения вводить не обязательно. В vim возможен повторный ввод команды с использованием истории команд.

Для работы с историей использованных команд необходимо находясь в командном режиме с выбранной английской раскладкой клавиатуры ввести символ «:» и нажать кнопку «стрелка вверх». После символа «:» будет подставлена последняя из введенных команд. Продолжая нажимать кнопку «стрелка вверх» можно пролистать все команды вводимые ранее.

Программа vim может выполнять редактирование двоичных файлов. Редактирование двоичных файлов необходимо для модификации исполняемых файлов программ в двоичном виде или любых других двоичных файлов. Для открытия файла в двоичном виде используется ключ «-b».

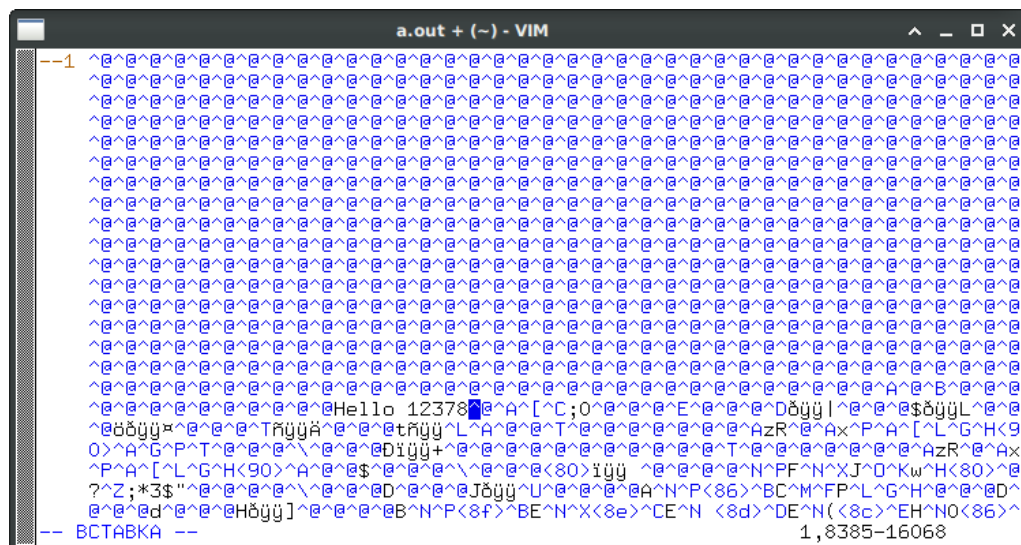
Используя исходный текст на языке Си представленный ниже, создан исполняемый файл с именем a.out.

```
1 #include <stdio.h>
2
3 int main() {
4     puts("Hello 12345");
5     return 0;
6 }
7
```

Для просмотра двоичного файла использована команда **vim -b a.out**.

После создания исполняемого файла выполнено открытие файла a.out, поиск в нем последовательности «12345», замена её на последовательность «12378» и сохранение изменений.

На рисунке представлено содержимое файла a.out и изменённая последовательность «12378».



В результате выполненных действий удалось изменить результат работы программы. На рисунке ниже в первой строке продемонстрирована строка запуска скомпилированной программы. На второй строке показан её вывод. Третья строка содержит команду открытия двоичного файл. Следует обратить внимание что ключ «-b» не указан, но vim, в данном случае, смог правильно

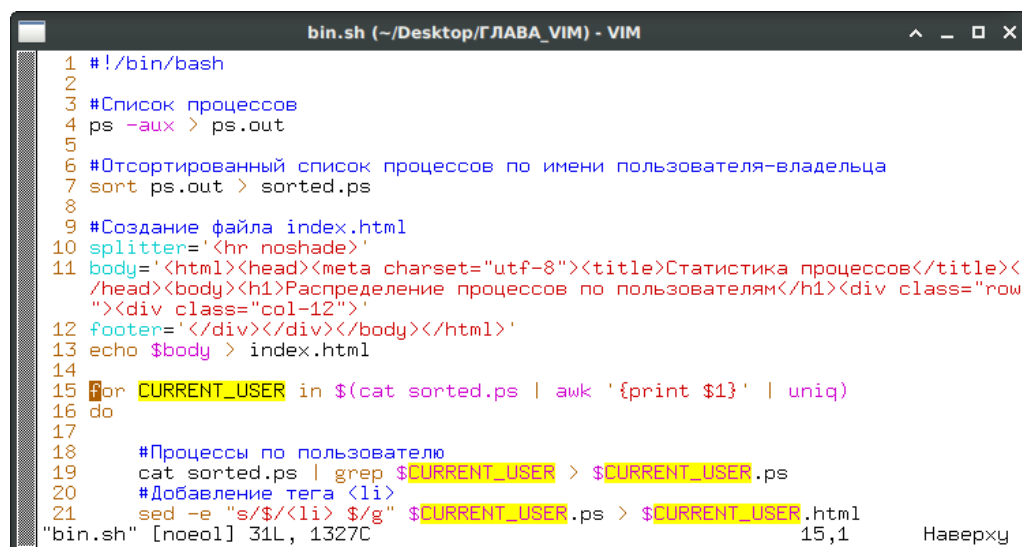
интерпретировать содержимое файла. В четвёртой строке выполнен повторный запуск изменённой программы, без повторной компиляции. В пятой строке выведена новая строка, отличающаяся на два последних символа. Таким образом, подтверждается возможность редактирования исполняемых файлов в двоичном виде без повторной компиляции.

```
[~]$ ./a.out
Hello 12345
[~]$ vim a.out
[~]$ ./a.out
Hello 12378
[~]$ █
```

При открытии текстового файла в vim возможен переход на нужную строку или функцию программы. На рисунке ниже первая строка показывает способ выполнения перехода на строку с номером 5. Вторая строка демонстрирует открытие файла с переходом на первое вхождение текстового шаблона «CURRENT_USER» в файле bin.sh.

```
]$ vim +5 bin.sh
]$ vim +/CURRENT_USER bin.sh
]$ █
```

На рисунке ниже показан результат открытия файла командой **vim +/CURRENT_USER bin.sh**.



```
bin.sh (~/.Desktop/ГЛАВА_VIM) - VIM
1  #!/bin/bash
2
3  #Список процессов
4  ps -aux > ps.out
5
6  #Отсортированный список процессов по имени пользователя-владельца
7  sort ps.out > sorted.ps
8
9  #Создание файла index.html
10 splitter='<hr noshade>'
11 body='<html><head><meta charset="utf-8"><title>Статистика процессов</title><
 /head><body><h1>Распределение процессов по пользователям</h1><div class="row
"><div class="col-12">'
12 footer='</div></div></body></html>'
13 echo $body > index.html
14
15 for CURRENT_USER in $(cat sorted.ps | awk '{print $1}' | uniq)
16 do
17
18     #Процессы по пользователю
19     cat sorted.ps | grep $CURRENT_USER > $CURRENT_USER.ps
20     #Добавление тега <li>
21     sed -e "s/</li> $/g" $CURRENT_USER.ps > $CURRENT_USER.html
"bin.sh" [noeol] 31L, 1327C                               15,1      Наверху
```

В дальнейшем подробное графическое представление результата ввода команд в редактор vim приводится не будет.

Редактор vim имеет множество **команд перемещения**, часть из которых не встречается в оконных текстовых редакторах. Для перемещения используется командный режим. Перемещение на один символ производится расположенными по порядку клавишами $\leftarrow \mathbf{h} \downarrow \mathbf{j} \uparrow \mathbf{k} \rightarrow \mathbf{l}$. Кнопки — стрелочки позволяют перемещаться в окне. Присутствует поддержка манипулятора «мышь».

Кроме перемещения на один символ возможны следующие варианты перемещения:

- на конец текущего слова вправо «**w**» или влево «**b**»;
- на следующее слово вправо «**W**» или влево «**B**»;
- в конец следующего слова «**E**»;
- на начало «**^**» или конец строки «**\$**»;
- на начало текущего предложения «**)**» или его конец «**(**»;
- на начало текущего абзаца «**{**» или его конец «**}**»;
- на первую строку на экране «**H**» или на последнюю «**L**»;
- переместится в конец текущего отображения буфера «**G**»;
- на первое вхождение произвольного символа «**f****v** — переход на первый символ **v** в строке»;
- повторить последнюю команду поиска в пределах текущей строки «**;**».

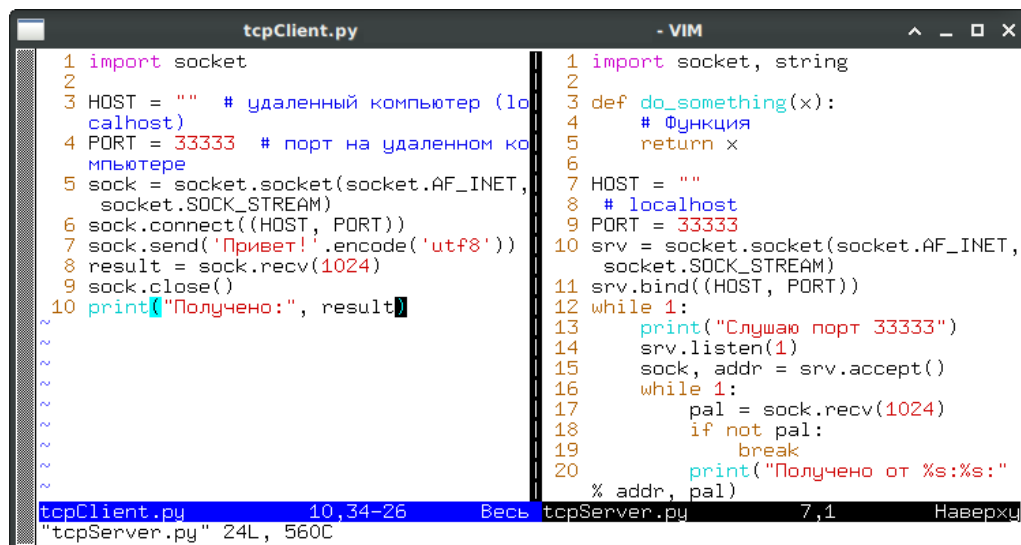
Почти все команды перемещения могут быть повторены произвольное количество раз. Например, **5j** переведёт курсор на 5 строк вверх, а **3}** — на три абзаца вниз. Этот способ повторения выполнения команды доступен для большинства команд других типов.

Рассмотрение **команд изменения текста** начнётся с команды «**u**» — отмена последнего действия. Набор команд редактирования наиболее удобен для написания исходного текста программы или для редактирования конфигурационного файла и также содержит команды, не встречающиеся в

графических текстовых редакторах. Далее приведены команды редактирования текста:

- если vim запущен в эмуляторе терминала, вставка из буфера обмена ОС «**ctrl+chift+v**»;
- редактирование одного текущего символа «**r**»;
- вставить символы с позиции курсора, заменив текущий символ под курсором «**i**»;
- добавить текст, начиная с позиции курсора «**a**»;
- заменить текст с начала строки «**I**»;
- вставить строку до текущей и приступить к её редактированию «**O**»;
- вставить строку после текущей и приступить к её редактированию «**o**»;
- удалить текущий символ «**d + стрелка вправо/стрелка влево**»;
- удалить текущее слово «**daw**»;
- удалить текущую строку «**dd**». Команда, удаляющая 10 строк **10dd**;
- удалить символы, начиная от курсора до конца строки «**D**»;
- «*****» - выделить текущее слово;
- «**/y клавиша Enter**» – подсветит все вхождения символа **y** в текущем окне. Перемещаться между выделенными символами можно с помощью клавиш «**N**» и «**n**»;
- «**y**» – копирование в регистр по умолчанию выделенного текста;
- «**yy**» или «**Y**» - копирование всей строки в регистр по умолчанию;
- «**v**» – режим «визуального выделения». До нажатия клавиши необходимо установить курсор в начало копирования, затем нажать клавишу **v**, далее, перемещаясь по тексту, выделить необходимый фрагмент текста и нажать клавишу «**y**»;
- «**/шаблон**» — поиск последовательности символов **шаблон**;
- «**p**» – вставка текста из регистра по умолчанию;
- «**%s/шаблон/замена**» – заменяет все вхождения слова **шаблон** на слово **замена**.

Несмотря на то, что программа vim не имеет графического интерфейса, она способна отображать одновременно буферы нескольких файлов. Команда **vim -o filename1 filename2** откроет два файла, разделённых по горизонтали. Команда **vim -O filename1 filename2** откроет эти же файлы, разделённые по вертикали. Ниже представлен результат выполнения команды **vim -O tcpClient.py tcpServer.py**. Данная команда сработает, если в текущем каталоге есть два файла tcpClient.py и tcpServer.py.



The screenshot shows the Vim editor interface with two buffers open side-by-side. The left buffer is titled 'tcpClient.py' and contains the following code:

```
1 import socket
2
3 HOST = "" # удаленный компьютер (localhost)
4 PORT = 33333 # порт на удаленном компьютере
5 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6 sock.connect((HOST, PORT))
7 sock.send('Привет!'.encode('utf8'))
8 result = sock.recv(1024)
9 sock.close()
10 print("Получено:", result)
```

The right buffer is titled '- VIM' and contains the following code:

```
1 import socket, string
2
3 def do_something(x):
4     # Функция
5     return x
6
7 HOST = ""
8 # localhost
9 PORT = 33333
10 srv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11 srv.bind((HOST, PORT))
12 while 1:
13     print("Слушаю порт 33333")
14     srv.listen(1)
15     sock, addr = srv.accept()
16     while 1:
17         pal = sock.recv(1024)
18         if not pal:
19             break
20         print("Получено от %s:%s:" % addr, pal)
```

At the bottom of the Vim window, the status bar shows the current buffer and cursor position: 'tcpClient.py 10,34-26' and 'tcpServer.py 7,1'.

Далее приведены команды для перехода между окнами:

- «:n» – переход в окно предыдущего буфера;
- «:N» – переход в окно следующего буфера;
- «ctrl + w h» - переход в окно влево;
- «ctrl + w l» - переход в окно справа;
- «ctrl + w k» - переход в окно сверху;
- «ctrl + w j» - переход в окно снизу.

Последние 4 комбинации вводятся как комбинация горячих клавиш. Способ их ввода отличается от способа нажатия горячих клавиш в программах с оконным интерфейсом. Данные комбинации горячих клавиш вводятся так:

1. одновременно нажимаются клавиши **ctrl** и **w**;

2. сразу после этого нажимается, например, клавиша **k** для перехода в верхнее окно, если оно есть.

Кроме буферов, vim оперирует понятием вкладки. Команда **vim -p10** откроет 10 пустых вкладок. Максимальное количество открываемых вкладок по умолчанию 10, однако может настраиваться с помощью файла конфигурации. Открытие нескольких файлов во вкладки выполняется командой **vim -p filename0 filename1 filename2**. Для переключения между вкладками используются команды:

- «**:tabedit filename**» – открытия файла в новую вкладку;
- «**:tabs**» – просмотр всех открытых вкладок;
- «**:tabn**» – переход на предыдущую вкладку;
- «**:tabp**» – переход на следующую вкладку;
- «**:tabfirst**» – переход на первую вкладку;
- «**:tablast**» – переход на конечную вкладку;
- «**:tabm 2**» – переход на третью вкладку, потому что нумерация вкладок начинается с 0.

Следует обратить внимание на следующую зависимость. Буфер отображается в окно, а окна располагаются во вкладках.

Программа vim способна выполнять сравнение файлов и выделять отличия цветом. При этом возможно одновременное сравнение двух и более файлов. Например, для сравнения трех файлов необходимо выполнить команду **vim -d filename1 filename2 filename3**. После этого они будут открыты с разделением по горизонтали и подсветкой отличий друг от друга. На рисунке ниже приведён пример открытия двух файлов для сравнения текста в них.

- **nerd tree** – навигация по файлам и каталогам, позволяет открыть просмотра файлов командой **NERDTree**;

- **nerd commenter** – упрощает добавления комментариев в исходный код программы;

- **ctrlp** – выполняет поиск файла по названию. Для исполнения необходимо нажать комбинацию клавиш ctrl+p и набрать имя файла;

- **vim airline** – повышение наглядности с помощью строки статуса программы vim;

- **taglist** – позволяет просматривать структуру программных файлов, список функций и прочую информацию;

- **supertab** – автодополнение слов нажатием клавиши tab;

- **project** – позволяет организовывать файлы в проекты.

Использование редактора vim в качестве основного инструмента поднимает престиж программиста среди разработчиков, даёт гибкий инструмент для создания программ с минимальным потреблением ресурсов.

Редактор vim можно использовать для форматирования текстовых файлов как стороннюю программу.

Далее приведён пример изменения кодировки в текстовом файле с помощью редактора vim.

Открыть файл filename в программе vim.

vim filename

Изменить кодировку для отображения имеющегося содержимого файла.

:e ++enc=cp1251

Изменить кодировку текста открытого файла.

:set fileencoding=utf-8

Задать формат последовательности перехода на новую строку (доступные значения dos, unix, mac).

:set fileformat=unix

Сохранение файла в новой кодировке.

:wq

Текстовый редактор vim является полнофункциональным редактором с большим набором функций. Команды vim имеют как полные, так и сокращённые формы записи. Обычно изучение возможностей vim происходит в процессе работы с ним.

Контрольные вопросы:

1. Как воспользоваться историей команд vim?
2. Как реализована возможность редактирования двоичных файлов в редакторе vim?
3. Перечислите несколько команд перемещения по текстовому документам в vim.
4. Как осуществлять поиск текста по шаблону в редакторе vim?
5. Как выполнить пятикратное удаление символа одной командой?
6. Как открыть два файла с вертикальным разделением?
7. Как изменить кодировку файла win1251 на utf-8 с помощью vim?
8. Чем отличаются буфера, окна и вкладки в vim?

Выполнить задания:

1. Набрать 15 строк исходного текста программы на языке Питон 3 используя редактор vim.
2. Удалить одной командой 10 строк набранного текста.
3. Вернуть удалённые строки..
4. В 10 строке вставить новую строку.
5. Удалить 8 строку.
6. Не закрывая первый файл, открыть второй файл в vim.
7. Сохранить изменения в первом файле и выйти из программы.
8. Подготовится продемонстрировать основные команды работы в vim.