Vim (Русский)

Состояние перевода: На этой странице представлен перевод статьи **Vim**. Дата последней синхронизации: 2 февраля 2018. Вы можете **помочь** синхронизировать перевод, если в английской версии произошли **изменения (https://wiki.archlinux.org/index.php?title=Vim&diff=0&oldid=509305)**.

Related articles

List of applications/Document

Vim - консольный текстовый редактор, являющийся расширенной версией **vi** с дополнительными функциями, которые включают в себя: подсветку синтаксиса, полноценную систему помощи, встроенную поддержку скриптов (vimscript), визуальный режим для простоты выделения и сравнение файлов (vimdiff).

Contents

Установка

Установите один из следующих пакетов:

- `vim (https://www.archlinux.org/packages/?name=vim)` — с поддержкой Python 2/3, Lua, Ruby и Perl, но без поддержки GTK/X.
- `gvim (https://www.archlinux.org/packages/?name=gvim)` — идентичен `vim`, но с поддержкой GTK/X.

Примечание:

- Пакет `vim` собран без поддержки **Xorg**; отсутствует компонент `+clipboard`, поэтому Vim не сможет оперировать с основным и **обменным буфером**. Пакет `gvim (https://www.archlinux.org/packages/?name=gvim)` предоставляет доступ также и к Vim с интерфейсом командной строки, но с поддержкой `+clipboard`
- Неофициальный репозиторий **herecura** предоставляет несколько версий Vim/gVim: `vim-cli`, `vim-gvim-common`, `vim-gvim-gtk`, `vim-gvim-qt`, `vim-rt` and `vim-tiny`.

Использование

Также вы можете запустить *vimtutor* (для версии с интерфейсом командной строки) или *gvimtutor* (для версии с графическим интерфейсом) для отображения руководства по использованию Vim.

Vim включает в себя широкую справочную систему, доступ к которой можно получить посредством `:h` *субъект*. Субъекты включают в себя команды, опции, горячие клавиши, плагины и так далее. Используйте команду `:h` (без какого-либо субъекта), чтобы полчуить информацию о вспомогательной системе и о том, как перемещаться между субъектами.

Настройка

Файл конфигурации Vim для конкретного пользователя расположен в домашней директории - `~/.vimrc`, файлы текущего пользователя в `~/.vim/`. Общий файл конфигурации - `~/.vimrc`, а общие файлы находятся в `/usr/share/vim/`.

Примечание: Более ли менее стандартное поведение - например, подсветка синтаксиса - определяется в `defaults.vim`, который загружается, когда отсутствует `~/.vimrc`. Добавьте `let skip_defaults_vim=1` в `/etc/vimrc` для полного отключения загрузки `defaults.vim`. **[1] (https://github.com/vim/vim/issues/1033)**

Буфер обмена

Такие команды Vim как `:yank` или `:paste` работают с безымянным регистром, который по умолчанию соответствует регистру `"*`. Если доступна `+clipboard`, то регистр `"*` отображается в `PRIMARY` буфер в X.

Чтобы изменить стандартный регистр на `"+`, используйте `:set clipboard=unnamedplus`. Регистр `"+` соответствует `CLIPBOARD` буферу в X.

Для более подробной информации смотрите `:help 'clipboard'`.

Совет: Для копирования и вставки могут быть созданы горячие клавиши. Смотрите, например, **[2] (http://superuser.com/a/189198)** для `ctrl+c`, `ctrl+v` и `ctrl+x`.

Подсветка синтаксиса

Чтобы включить подсветку синтаксиса (Vim поддерживает огромный список языков программирования):

```
:filetype plugin on
:syntax on
```

Визуальный перенос

Опция `wrap` (включена по умолчанию) указывает Vim переносить длинные строки, которые не помещаются на экран, так, что оставшаяся часть строки отображается на следующей линии. Опция `wrap` влияет только на отображение текста, сам текст при этом не изменяется.

Изначально перенос строки происходит ровно в том месте, где размещается последний ее

символ, поместившийся на экране, даже если он находится в середине слова. Для более умного переноса используйте опцию `linebreak`. Когда эта опция включена командой `set linebreak`, перенос строки происходит только после символов, которые перечислены в опции `breakat`, которая по умолчанию содержит в себе пробел и некоторые знаки препинания (смотрите `:help breakat`).

Остаток строки обычно начинается с начала следующей линии, без всякого отступа. Опция **breakindent (https://retracile.net/wiki/VimBreakIndent)** сообщает Vim, что необходимо отображать перенесенный остаток строки с отступом, так, что на всех последующих линиях перенесенные части строки имеют тот же отступ, что и начало этой строки. Поведение `breakindent` может быть настроено при помощи опции `breakindentopt`. Например, для файлов исходного кода на языке Python может быть полезно добавлять дополнительный отступ в 4 пробела для перенесенных частей длинной строки (подробнее смотрите в `:help breakindentopt`):

```
autocmd FileType python set breakindentopt=shift:4
```

Использование мыши

Vim позволяет пользоваться мышью, но только в тех терминалах, которые это поддерживают:

- **xterm**/**urxvt**
- Консоль linux с `gpm (https://www.archlinux.org/packages/?name=gpm)` (смотрите подробнее на странице **Console mouse support**)
- **PuTTY**

Чтобы включить поддержку мыши, добавьте в `~/.vimrc`:

```
set mouse=a
```

Опция `mouse=a` устанавливается в `defaults.vim`

Примечание: Копирование/вставка будут использовать регистр `"*`, если есть доступ к X серверу; смотрите раздел **#Буфер обмена**. xterm по-прежнему может обрабатывать нажатия мыши, когда зажат `shift`.

Переход на новую строку с помощью клавиш со стрелками

По умолчанию, при нажатии `←` в начале строки или `→` в конце, курсор не переводится на следующую/предыдущую строку.

Это можно исправить с помощью добавления строки `set whichwrap=b,s,<,>,[,]` в ваш файл `~/.vimrc`.

Объединение файлов

Vim включает в себя diff-редактор (программа, которая отображает различия между двумя файлами и позволяет удобно их объединять). Используйте vimdiff для этого — просто укажите ей пару файлов: `vimdiff` *`файл1 файл2`*. Ниже приведен список команд *vimdiff*.

Действие	Горячая клавиша
следующие изменение	`]c`
предыдущее изменение	`[c`
diff obtain	`do`
diff put	`dp`
развернуть блок	`zo`
свернуть блок	`zc`
перечитать файлы	`:diffupdate`
переключить окна	`Ctrl+w+w`

Советы и хитрости

Нумерация строк

Чтобы включить отображение номера столбца, используйте `:set number`. По умолчанию показываются абсолютные значения номеров строк, относительные включаются посредством `:set relativenumber`.

Переход к новой строке осуществляется посредством `:line number` или `line numbergg`. Все переходы запоминаются в лист переходов, для более подробной информации смотрите `:h jump-motions`.

Проверка орфографии

Проверку орфографии в Vim можно включить с помощью:

```
set spell
```

По умолчанию установлен только английский словарь. Другие словари можно найти в **официальных репозиториях** по запросу `vim-spell`. Еще больше словарей можно найти в **FTP-архиве Vim (http://ftp.vim.org/vim/runtime/spell/)**. Словари необходимо поместить в каталог для словарей — `~/.vim/spell/`. Включить словарь можно командой `:setlocal spell spelllang=ru_yo` (заменив `ru_yo` на имя нужного словаря).

Действие	Команда
следующая ошибка	`]s`
предыдущая ошибка	`[s`
предложения для исправления	`z=`
добавить правильное написание	`zg`
добавить правильное написание (на сеанс)	`zG`
добавить неправильное написание	`zw`
добавить неправильное написание (на сеанс)	`zW`
повторить проверку орфографии во всем файле	`:spellr`

Совет:

- Чтобы включить проверку сразу для двух языков (например, английского и русского), добавьте `set spelllang=en,ru` в `~/.vimrc` или `/etc/vimrc` и перезапустите Vim.
- Вы можете включить проверку орфографии для конкретных типов файлов (например *.txt*), используя плагин FileType и собственное правило для определения типа файла. Чтобы включить проверку орфографии для всех файлов, оканчивающихся на *.txt*, создайте файл `/usr/share/vim/vimfiles/ftdetect/plaintext.vim` и вставьте туда строку: `autocmd BufRead,BufNewFile *.txt setfiletype plaintext`. Далее, вставьте строку `autocmd FileType plaintext setlocal spell spelllang=ru` в файл `~/.vimrc` или `/etc/vimrc` и перезапустите Vim.
- Чтобы включить проверку орфографии только для документов LaTeX (или TeX), добавьте `autocmd FileType **tex** setlocal spell spelllang=ru` в файл `~/.vimrc` или `/etc/vimrc` и перезапустите Vim.

Сохранение позиции курсора

Если вы хотите, чтобы курсор **возвращался в прежнее положение (http://vim.wikia.com/wiki/Restore_cursor_to_file_position_in_previous_editing_session)** после открытия файла, добавьте следующее в `~/.vimrc`:

```
augroup resCur
  autocmd!
  autocmd BufReadPost * call setpos(".", getpos("'\""))
augroup END
```

Запуск Vim по команде vi

Создайте **псевдоним** для `vi` на `vim`.

Если вы хотите, чтобы при вводе `sudo vi` запускался `vim`, то **установите** пакет **vi-vim-symlink (https://aur.archlinux.org/packages/vi-vim-symlink/)**[AUR], который удалит `vi` и заменить его символической ссылкой на `vim`.

Возврат каретки DOS/Windows

Если вы видите последовательность `^M` в конце каждой линии, это означает, что вы редактируете текстовый файл, который был создан в MS-DOS или Windows. Дело в том, что в Linux для переноса принято использовать один символ новой строки (LR), тогда как в системах Windows/MS DOS для той же цели используется последовательность из двух символов: возврата каретки (CR) и новой строки (LR). Как раз эти символы возврата каретки и отображаются в виде `^M`.

Для удаления всех символов возврата каретки из файла, выполните:

```
:%s/^M//g
```

Обратите внимание, что `^` здесь — управляющий символ, а не обычный. Чтобы ввести управляющую последовательность `^M`, нажмите `Ctrl+v,Ctrl+m`.

Также вы можете просто установить пакет **dos2unix (https://www.archlinux.org/packages/?name=dos2unix)** и исправлять файлы командой `dos2unix` *`файл`*.

Примечание: Другой простой способ заключается в изменении опции `fileformat`. Используйте `set ff=unix` для преобразования файлов с окончанием строки DOS/Windows в файлы с окончанием строки Unix. Для того, чтобы сделать обратное, просто используйте `set ff=dos`.

Пустое пространство в нижней части окон gVim

Когда используется **оконный менеджер**, настроенный на игнорирование размеров окна, gVim заполняет неиспользованные области окна стандартным фоновым цветом темы GTK.

Решение заключается в регулировании количества места, которое будет резервировать в нижней части окна gVim. Поместите следующую строку в `~/.vimrc`:

```
set guiheadroom=0
```

Примечание: Установив значение 0, вы не сможете видеть горизонтальную полосу прокрутки внизу окна.

Плагины

Плагины могут помочь повысить эффективность вашей работы в Vim. Они могут менять интерфейс Vim, добавлять новые команды, поддержку завершения кода, интеграцию других программ и утилит с Vim, поддержку дополнительных языков и многое другое.

Совет: For a list of popular plugins, see Vim Awesome)

Установка

Используя встроенную систему управления пакетами

В Vim 8 добавлена возможность загрузки сторонних плагинов. Чтобы использовать эту функциональность, переместите плагины в `~/.vim/pack/foo`.

Используя менеджер плагинов

Менеджер плагинов позволяет устанавливать плагины и управлять ими одинаковым образом независимо от того, на какой системе вы запускаете Vim. Он представляет собой специальный плагин, который выполняет роль пакетного менеджера для других плагинов.

- **Vundle (https://github.com/gmarik/Vundle.vim)** — в настоящее время наиболее популярный менеджер плагинов Vim.
- **Vim-plug (https://github.com/junegunn/vim-plug)** - минималистичный менеджер плагинов для Vim, который имеет много особенностей, например, загрузка плагинов по запросу и параллельное обновление.
- **pathogen.vim (https://github.com/tpope/vim-pathogen)** — простой плагин для управления переменной runtimepath.
- **Dein.vim (https://github.com/Shougo/dein.vim)** - менеджер плагинов, заменяющий **NeoBundle (https://github.com/Shougo/neobundle.vim)**, доступен как `vim-dein-git (https://aur.archlinux.org/packages/vim-dein-git/)`[AUR].

Из репозиториев Arch

Группа `vim-plugins (https://www.archlinux.org/groups/x86_64/vim-plugins/)` объединяет множество разнообразных плагинов. Используйте команду `pacman -Sg vim-plugins` для отображения списка пакетов, которые вы можете затем **установить**, используя pacman.

cscope

Cscope (http://cscope.sourceforge.net/) является инструментом для навигации по программному проекту. Путем перехода к слову/символу/функции и вызову cscope (обычно с помощью горячих клавиш) можно найти: функции, которые вызывают данную, определение функции и многое другое.

Установите пакет `cscope (https://www.archlinux.org/packages/?name=cscope)`.

Скопируйте стандартный файл cscope, который Vim будет автоматически читать:

```
mkdir -p ~/.vim/plugin
wget -P ~/.vim/plugin http://cscope.sourceforge.net/cscope_maps.vim
```

Примечание: Вам наверное понадобится откомментировать следующие строки в `~/.vim/plugin/cscope_maps.vim`, чтобы включить горячие клавиши cscope в Vim 7.x:

```
set timeoutlen=4000
set ttimeout
```

Создайте файл, который содержит список файлов для индексации (cscope может обрабатывать множество языков, но для примера мы возьмем файлы на Си/Си++ с расширениями *.c*, *.cpp* и *.h*):

```
cd /путь/к/каталогу/проекта
find . -type f -print | grep -E '\.(c(pp)?|h)$' > cscope.files
```

Создание файлы базы данных, которые cscope будет читать:

```
cscope -bq
```

Примечание: cscope ищет файл `cscope.out` в рабочем каталоге, поэтому тот же каталог следует использовать и для навигации по проекту в Vim. Также, вы можете указать путь до этого файла явно, установив его в переменной окружения `$CSCOPE_DB`.

Горячие клавиши по умолчанию:

```
Ctrl-\ и
     c: найти функции вызывающие эту функцию
```

```
    d: найти функции вызываемые этой функцией
    e: найти egrep паттерн
    f: найти этот файл
    g: найти это определение
    i: найти файлы, включающие этот файл (#include)
    s: найти этот символ Си
    t: найти присвоения
```

Не стесняйтесь изменять горячие клавиши:

```
#Maps ctrl-c to find functions calling the function
nnoremap <C-c> :cs find c <C-R>=expand("<cword>")<CR><CR>
```

Taglist

Taglist (http://vim-taglist.sourceforge.net/) отображает структуру файлов исходного кода и позволяет эффективно просматривать исходники на различных языках программирования.

Установите пакет `vim-taglist (https://www.archlinux.org/packages/?name=vim-taglist)`.

Полезные опции можно записать в `~/.vimrc`:

```
let Tlist_Compact_Format = 1
let Tlist_GainFocus_On_ToggleOpen = 1
let Tlist_Close_On_Select = 1
nnoremap <C-l> :TlistToggle<CR>
```

Смотрите также

Официальные ресурсы

- **Домашняя страница (http://www.vim.org/)**
- **Документация (http://vimdoc.sourceforge.net/)**
- **Vim Wiki (http://vim.wikia.com)**
- **Vim Scripts (http://www.vim.org/scripts/)**

Руководства

- **vim Tutorial and Primer (https://danielmiessler.com/study/vim/)**
- **vi Tutorial and Reference Guide (http://usalug.org/vi.html)**
- **Graphical vi-Vim Cheat Sheet and Tutorial (http://www.viemu.com/a_vi_vim_graphical_cheat_sheet_tutorial.html)**
- **Vim Introduction and Tutorial (http://blog.interlinked.org/tutorials/vim_tutorial.html)**
- **Open Vim (http://www.openvim.com/)** - Коллекция средств обучения Vim.
- **Learn Vim Progressively (http://yannesposito.com/Scratch/en/blog/Learn-Vim-Progressively/)**
- **Learning Vim in 2014 (http://benmccormick.org/learning-vim-in-2014/)**
- **Seven habits of effective text editing (http://www.moolenaar.net/habits.html)**
- **Basic Vim Tips (http://bencrowder.net/files/vim-fu/)**
- **HOWTO Vim (http://www.gentoo-wiki.info/HOWTO_VIM)**

Видео

- **Vimcasts (http://vimcasts.org/)** - Скринкаст в формате .ogg.
- **Vim Tutorial Videos (http://derekwyatt.org/vim/tutorials/)** - От основ к продвинутым темам.

Игры

- **Vim Adventures (http://vim-adventures.com/)**
- **VimGolf (http://vimgolf.com/)**

Конфигурация

- **nion (https://web.archive.org/web/20131020125020/http://nion.modprobe.de/setup/vimrc)**
- **Детальная конфигурация от Amir Salihefendic (http://amix.dk/vim/vimrc.html)**
- **Bart Trojanowski (https://web.archive.org/web/20131004071740/http://www.jukie.net/~bart/conf/vimrc)**
- **Steve Francia's Vim Distribution (https://github.com/spf13/spf13-vim)**
- **Vim Awesome (http://vimawesome.com/)** - Vim Plugins
- **Конфигурация Vim W4RH4W (https://github.com/W4RH4WK/dotVim)**
- **Fast vimrc/colorscheme from askapache (https://www.askapache.com/linux/fast-vimrc/)**
- **Базовый vimrc (https://gist.github.com/anonymous/c966c0757f62b451bffa)**
- **Usevim (http://www.usevim.com/)**

Цвета

- **Vivify (http://bytefluent.com/vivify/)**
- **Vim colorscheme customization (https://linuxtidbits.wordpress.com/2014/10/14/vim-customize-installed-colorschemes/)**

Retrieved from "https://wiki.archlinux.org/index.php?title=Vim_(Русский)&oldid=511807"
