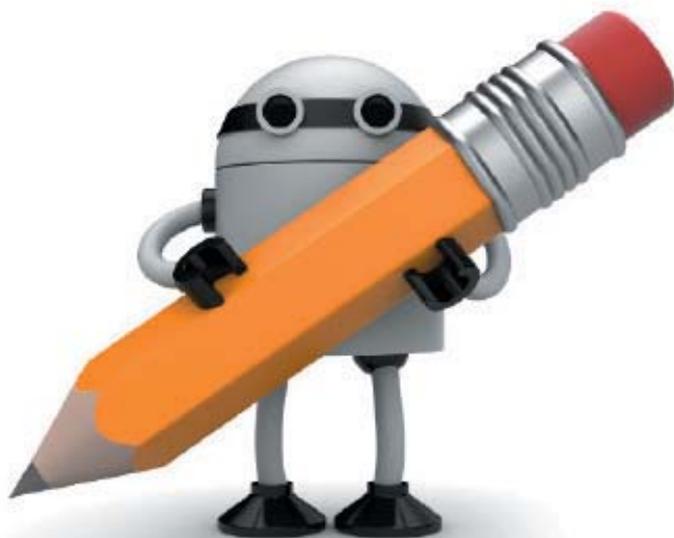


The  
Pragmatic  
Programmers

# Практическое использование Vim

Редактируйте текст  
со скоростью мысли



Дрю Нейл



**УДК 004.912Vim**  
**ББК 32.973.26-018.2**  
**Н38**

Нейл Д.  
Н38 Практическое использование Vim. – М.: ДМК Пресс, 2014. – 384 с.: ил.

ISBN 978-5-94074-972-1

Vim – быстрый и эффективный текстовый редактор, способный повысить скорость и эффективность разработки. С помощью более чем 100 рецептов вы быстро освоите основные возможности Vim и сможете заняться решением своих самых необычных задач, связанных с созданием и правкой текста.

В данной книге вы найдете новые и эффективные приемы работы с редактором, независимо от того, являетесь ли вы начинающим или опытным пользователем Vim.

**УДК 004.912Vim**  
**ББК 32.973.26-018.2**

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-934356-98-2 (анг.)

Copyright © 2012 The Pragmatic  
Bookshelf

ISBN 978-5-94074-972-1 (рус.)

© Оформление, перевод,  
ДМК Пресс, 2014



# Содержание

<b>Благодарности</b> .....	20
<b>Предисловие</b> .....	23
<b>Это надо знать</b> .....	25
<b>Прочитайте всеми забытое руководство</b> .....	28
<b>Глава 1. Путь Vim</b> .....	39
Рецепт 1. Встречайте: команда «точка» .....	39
Команда «точка» – микромакроопределение .....	42
Рецепт 2. Не повторяйся .....	42
Избавляйтесь от лишних перемещений .....	43
Рецепт 3. Шаг назад, три вперед .....	45
Делайте изменения повторяемыми .....	45
Делайте перемещения повторяемыми .....	46
Теперь все вместе .....	46
Рецепт 4. Действие, повтор, возврат .....	46
Рецепт 5. Поиск и замена вручную .....	48
Будьте экономны: выполняйте поиск без ввода лишних символов .....	49
Делайте изменения повторяемыми .....	50
Теперь все вместе .....	50
Рецепт 6. Формула точки .....	50
Обзор решений трех задач редактирования с помощью команды «точка» .....	51
Идеальное решение: одна клавиша для перехода и одна для изменения .....	51
<b>Часть I. РЕЖИМЫ</b> .....	52
<b>Глава 2. Командный режим</b> .....	53
Рецепт 7. Оторвите кисть от холста .....	53
Рецепт 8. Группируйте изменения для возможной отмены .....	54
Рецепт 9. Составляйте повторяемые изменения .....	56
Удаление назад .....	56
Удаление вперед .....	57
Удаление целого слова .....	57

Дополнительная оценка: какой вариант более повторим? ..	57
Обсуждение.....	58
Рецепт 10. Используйте счетчики для простых арифметических операций.....	58
Рецепт 11. Не занимайтесь подсчетами, если можно выполнить повторение .....	60
Используйте счетчик, когда в этом есть смысл.....	62
Рецепт 12. Объединяй и властвуй.....	63
Оператор + команда перемещения = Действие.....	63
Расширение возможностей Vim .....	64
<b>Глава 3. Режим вставки .....</b>	<b>67</b>
Рецепт 13. Исправляйте ошибки, не выходя из режима вставки.....	67
Рецепт 14. Возвращайтесь в командный режим .....	68
Встречайте: командный подрежим режима вставки.....	69
Рецепт 15. Вставка из регистра, не покидая режима вставки .....	70
Используйте <C-r>{register} для доступа к регистрам .....	71
Рецепт 16. Выполняйте простые вычисления на месте .....	72
Рецепт 17. Вставка необычных символов по их кодам .....	73
Рецепт 18. Вставка необычных символов, соответствующих парам символов .....	74
Рецепт 19. Правка текста в режиме замены.....	74
Затирайте символы табуляции в виртуальном режиме замены .....	75
<b>Глава 4. Визуальный режим .....</b>	<b>77</b>
Рецепт 20. Знакомство с визуальным режимом.....	77
Рецепт 21. Выделение текста в визуальном режиме .....	79
Включение визуальных режимов.....	80
Переключение между визуальными режимами .....	80
Переключение свободного конца выделения .....	81
Рецепт 22. Повторение команд построчного визуального режима .....	81
Подготовка .....	82
Выполните отступ один раз, а затем повторите .....	82
Рецепт 23. Используйте операторы вместо команд визуального режима, если это возможно .....	84
Использование визуального оператора .....	84

Использование операторов командного режима .....	85
Обсуждение.....	85
Рецепт 24. Правка табличных данных в блочном визуальном режиме .....	86
Рецепт 25. Изменение колонок текста .....	88
Рецепт 26. Добавление текста после прямоугольного блока ...	90
<b>Глава 5. Режим командной строки.....</b>	<b>92</b>
Рецепт 27. Встречайте: режим командной строки .....	92
Специальные ключи в режиме командной строки .....	94
Команды Ex стреляют дальше и накрывают большую площадь .....	95
Рецепт 28. Выполнение команд для одной строки или для группы смежных строк .....	96
Номера строк.....	96
Определяйте диапазон строк с использованием их номеров ...	97
Определяйте диапазон строк посредством визуального выделения .....	98
Определяйте диапазон строк с помощью шаблонов .....	99
Изменяйте адрес с помощью смещения .....	100
Обсуждение.....	100
Рецепт 29. Копируйте и перемещайте строки с помощью команд ':t' и ':m' .....	101
Копируйте строки командой :t.....	101
Перемещайте строки командой ':m' .....	103
Рецепт 30. Применение команд командного режима к диапазону строк.....	104
Рецепт 31. Повторяйте последнюю команду Ex .....	106
Рецепт 32. Автодополнение команд Ex .....	108
Выбор из нескольких совпадений .....	109
Рецепт 33. Вставка текущего слова в командную строку.....	110
Рецепт 34. Повторный вызов команд из истории .....	111
Встречайте: окно режима командной строки .....	112
Рецепт 35. Выполнение команд в оболочке .....	115
Запуск программ в командной оболочке.....	115
Передача содержимого буфера на вход командам или сохранение вывода команд в буфере .....	117
Фильтрация содержимого буфера с помощью внешней команды .....	118
Обсуждение.....	119

<b>Часть II. ФАЙЛЫ</b> .....	121
<b>Глава 6. Управление несколькими файлами</b> .....	122
Рецепт 36. Слежение за открытыми файлами с помощью списка буферов .....	122
Различия между файлами и буферами .....	122
Встречайте: список буферов .....	123
Пользуйтесь списком буферов .....	124
Удаление буферов .....	125
Рецепт 37. Группировка буферов в коллекции с помощью списка аргументов .....	126
Заполнение списка аргументов .....	127
Определение файлов по именам .....	127
Определение шаблонов имен файлов .....	127
Определение файлов с помощью обратных кавычек .....	128
Использование списка аргументов .....	129
Рецепт 38. Управление скрытыми файлами .....	129
Обработка скрытых буферов при выходе из редактора .....	131
Включите параметр настройки hidden перед вызовом команды :argdo или :bufdo .....	131
Рецепт 39. Деление рабочего пространства на окна .....	133
Создание окон .....	133
Переключение фокуса ввода между окнами .....	134
Закрытие окон .....	135
Изменение размеров и переупорядочение окон .....	135
Рецепт 40. Организация размещения окон с помощью вкладок .....	136
Как пользоваться вкладками .....	137
Открытие и закрытие вкладок .....	138
Переключение между вкладками .....	138
Переупорядочение вкладок .....	139
<b>Глава 7. Открытие файлов и их сохранение на диск</b> ....	140
Рецепт 41. Открытие файла по его имени с использованием команды :edit .....	140
Как открыть файл, указав путь относительно текущего рабочего каталога .....	141
Как открыть файл, указав путь относительно активного каталога .....	142

Рецепт 42. Открытие файла по его имени с применением команды :find.....	143
Подготовка .....	143
Настройка параметра path.....	144
Используйте команду :find для поиска файлов по именам ...	145
Рецепт 43. Исследование файловой системы с помощью netrw .....	145
Подготовка .....	146
Встречайте: netrw – встроенный обозреватель файлов Vim... ..	146
Открытие обозревателя файлов .....	147
Работа с окнами.....	148
Дополнительные возможности netrw .....	149
Рецепт 44. Сохранение файлов в несуществующие каталоги ...	150
Рецепт 45. Сохранение файла с привилегиями суперпользователя .....	150
<b>Часть III. БЫСТРАЯ НАВИГАЦИЯ .....</b>	<b>153</b>
<b>Глава 8. Навигация внутри файлов .....</b>	<b>154</b>
Рецепт 46. Не убирайте руки из основной позиции.....	155
Оставьте свою правую руку там, где она должна быть .....	156
Рецепт 47. Разница между фактическими строками и строками на экране .....	157
Рецепт 48. Перемещение по словам.....	159
Отличайте слова и СЛОВА .....	161
Рецепт 49. Поиск символов .....	162
Поиск символа может выполняться включительно или исключительно .....	165
Думайте как при игре в «Балду» .....	166
Рецепт 50. Поиск с целью навигации .....	166
Используйте команды поиска в операциях.....	168
Рецепт 51. Выделение фрагментов с применением текстовых объектов .....	169
Выполнение операций с текстовыми объектами .....	172
Обсуждение.....	172
Рецепт 52. Удаление, включая ограничители, и изменение внутри ограничителей .....	173
Рецепт 53. Установка меток и возврат к ним .....	175
Автоматическая расстановка меток .....	176
Рецепт 54. Переход между парными скобками .....	177
Переход между парными ключевыми словами .....	178

<b>Глава 9. Навигация между файлами</b> .....	180
Рецепт 55. Обход списка переходов .....	180
Рецепт 56. Обход списка изменений .....	182
Отметка последнего изменения.....	183
Рецепт 57. Переход к файлу с именем под курсором .....	184
Определение расширения файла .....	185
Определение списка каталогов для поиска .....	185
Обсуждение.....	186
Рецепт 58. Переключение между файлами с помощью глобальных меток .....	187
Устанавливайте глобальную метку перед погружением в код .....	188
<b>Часть IV. РЕГИСТРЫ</b> .....	189
<b>Глава 10. Копирование и вставка</b> .....	190
Рецепт 59. Удаление, копирование и вставка с применением неименованного регистра .....	190
Перестановка символов .....	191
Перестановка строк.....	191
Создание дубликатов строк .....	192
Ой! Я затер свою копию .....	192
Рецепт 60. Знакомство с регистрами Vim .....	193
Адресация регистров.....	194
Неименованный регистр ("") .....	195
Регистр захвата («0»).....	196
Именованные регистры ("a–z") .....	196
Регистр «черной дыры» ("_").....	197
Системный буфер обмена ("+) и регистр выделенного фрагмента ("*") .....	198
Регистр выражений ("=) .....	199
Дополнительные регистры .....	199
Рецепт 61. Замена выделенного текста содержимым регистра.....	200
Как поменять слова местами .....	201
Рецепт 62. Вставка из регистра .....	202
Вставка последовательностей символов .....	202
Вставка строк .....	203
Обсуждение.....	204
Рецепт 63. Взаимодействие с системным буфером обмена....	204

Подготовка .....	205
Вызов системной команды вставки.....	205
Использование системной команды вставки в режиме вставки .....	206
Используйте регистр "+, чтобы исключить необходимость переключения параметра paste .....	207
<b>Глава 11. Макросы</b> .....	<b>208</b>
Рецепт 64. Запись и выполнение макроса .....	209
Запись последовательности команд в виде макроса .....	209
Повторное выполнение последовательности команд вызовом макроса.....	210
Последовательное выполнение макроса .....	211
Параллельное выполнение макроса .....	212
Рецепт 65. Товсь! Цельсь! Отставить! .....	212
Нормализация позиции курсора .....	212
Устанавливайте курсор повторяемыми командами перемещения .....	213
Используйте возможность прерывания при неудачном перемещении курсора .....	213
Рецепт 66. Выполнение со счетчиком.....	214
Рецепт 67. Повторение изменений в последовательности строка.....	216
Запись единицы правки .....	216
Последовательное выполнение макроса .....	218
Параллельное выполнение макроса .....	219
Выбор: последовательно или параллельно .....	220
Рецепт 68. Добавление команд в макросы.....	220
Обсуждение.....	221
Рецепт 69. Выполнение операций над множеством файлов ....	222
Подготовка .....	222
Создание списка целевых файлов .....	223
Запись единицы правки .....	223
Параллельное выполнение макроса .....	224
Последовательное выполнение макроса .....	225
Сохранение изменений во всех файлах .....	226
Обсуждение.....	226
Рецепт 70. Использование итератора для нумерации элементов в списке .....	227

Простой язык сценариев Vim .....	228
Запись макроса .....	228
Выполнение макроса .....	229
Рецепт 71. Правка содержимого макроса .....	229
Задача: Нестандартное форматирование .....	230
Вставка макроса в документ .....	231
Правка макроса .....	231
Копирование макроса из документа обратно в регистр ....	231
Обсуждение .....	232
<b>Часть V. ШАБЛОНЫ</b> .....	234
<b>Глава 12. Поиск по шаблону и поиск точного совпадения</b> .....	235
Рецепт 72. Настройка чувствительности к регистру в шаблонах .....	235
Глобальная настройка чувствительности к регистру .....	236
Настройка чувствительности к регистру для каждой операции поиска .....	236
Интеллектуальное определение чувствительности к регистру .....	236
Рецепт 73. Включение поддержки регулярных выражений ....	237
Поиск шестнадцатеричных кодов цвета в расширенном режиме .....	238
Поиск шестнадцатеричных кодов цвета в самом волшебном режиме .....	238
Использование класса шестнадцатеричных цифр .....	239
Обсуждение .....	239
Рецепт 74. Ключ \V включает режим поиска точного совпадения .....	240
Рецепт 75. Использование круглых скобок для захвата совпадений с подвыражениями .....	242
Рецепт 76. Границы слова .....	244
Рецепт 77. Границы совпадения .....	246
Рецепт 78. Экранирование проблемных символов .....	248
Экранируйте символы «/», выполняя поиск вперед .....	248
Экранируйте символы «?», выполняя поиск назад .....	249
Всегда экранируйте символы «\» .....	250
Экранируйте символы программно .....	251

<b>Глава 13. Поиск</b> .....	253
Рецепт 79. Встречайте: команда поиска .....	253
Выполнение команды поиска .....	254
Определение направления поиска .....	254
Повторение последней команды поиска .....	254
История поиска .....	255
Рецепт 80. Подсветка совпадений .....	256
Отключение подсветки совпадений .....	256
Рецепт 81. Предварительный просмотр первого совпадения .....	257
Проверка существования совпадения .....	258
Автодополнение поля ввода шаблона с опорой на предварительные результаты поиска .....	258
Рецепт 82. Подсчет совпадений с текущим шаблоном .....	259
Рецепт 83. Смещение курсора относительно конца совпадения .....	259
Рецепт 84. Выполнение операций над полным совпадением ...	262
Рецепт 85. Создание сложных шаблонов с использованием истории поиска .....	265
1: Максимальное совпадение .....	265
2: Доработка .....	266
3: Еще один цикл .....	267
4: Последний штрих .....	268
Обсуждение .....	268
Рецепт 86. Поиск текущего визуального выделения .....	269
Поиск текущего слова в визуальном режиме .....	269
Поиск текущего выделения (прототип) .....	270
Поиск текущего выделения (окончательная версия) .....	270
<b>Глава 14. Подстановка</b> .....	272
Рецепт 87. Встречайте: команда подстановки .....	272
Настройка поведения команды подстановки с помощью флагов .....	273
Специальные символы в строке замены .....	274
Рецепт 88. Поиск и замена всех совпадений в файле .....	274
Рецепт 89. Подтверждение каждой подстановки .....	276
Обсуждение .....	277
Рецепт 90. Повторное использование последнего шаблона поиска .....	278

Этот прием подходит не всегда.....	279
Влияние на историю команд .....	279
Рецепт 91. Замена содержимым регистра.....	280
Передача по значению .....	280
Передача по ссылке.....	281
Сравнение .....	281
Рецепт 92. Повторение предыдущей команды подстановки ....	282
Повторение команды подстановки в строке ко всему файлу .....	283
Изменение диапазона в команде подстановки.....	283
Обсуждение.....	285
Рецепт 93. Переупорядочение полей в файле CSV .....	286
Рецепт 94. Выполнение арифметических операций в строке замены .....	287
Шаблон поиска .....	288
Команда подстановки .....	288
Рецепт 95. Перемена местами двух и более слов .....	289
Возврат другого слова .....	290
Поиск совпадений с двумя словами.....	290
Все вместе .....	290
Обсуждение.....	291
Рецепт 96. Поиск и замена в нескольких файлах.....	292
Команда подстановки .....	292
Применение команды подстановки ко всем файлам в текущем проекте .....	293
Конструирование списка файлов, содержащих совпадения с шаблоном .....	294
<b>Глава 15. Глобализация команд .....</b>	<b>297</b>
Рецепт 97. Встречайте: команда :global .....	297
Рецепт 98. Удаление строк, соответствующих шаблону .....	298
Удаление соответствующих строк командой :g/re/d.....	299
Сохранение только соответствующих строк командой :v/re/d .....	300
Рецепт 99. Выборка комментариев TODO в регистр .....	300
Обсуждение.....	302
Рецепт 100. Сортировка свойств в правилах CSS .....	303
Сортировка свойств внутри одного блока .....	303
Сортировка свойств во всех блоках .....	304
Обсуждение.....	306

<b>Часть VI. ИНСТРУМЕНТЫ</b> .....	307
<b>Глава 16. Индексирование исходного кода и навигация по нему с помощью stags</b> .....	308
Рецепт 101. Встречайте: stags .....	308
Установка exuberant stags .....	308
Индексирование исходного кода проекта с помощью stags... ..	309
Анатомия индексного файла .....	310
Ключевые слова адресуются шаблонами, а не номерами строк .....	310
Индексирование ключевых слов с помощью метаданных ... ..	311
Рецепт 102. Настройка Vim для работы с программой stags... ..	311
Настройка поиска индексного файла в Vim .....	312
Создание индексного файла .....	312
Обсуждение .....	313
Рецепт 103. Навигация по определениям ключевых слов .....	314
Переход к определению ключевого слова .....	314
Определение точки перехода при наличии нескольких совпадений с ключевым словом .....	315
Использование команд Ex .....	317
<b>Глава 17. Компиляция кода и обзор ошибок с помощью Quickfix List</b> .....	319
Рецепт 104. Компиляция кода, не покидая Vim .....	320
Подготовка .....	320
Компиляция проекта в командной оболочке .....	320
Компиляция проекта в Vim .....	321
Рецепт 105. Навигация по списку с результатами .....	323
Основы навигации по списку с результатами .....	324
Быстрые переходы вперед и назад .....	324
Окно Quickfix .....	325
Рецепт 106. Восстановление прежнего списка с результатами .....	325
Рецепт 107. Настройка внешнего компилятора .....	326
Настройка вызова программы Nodelint командой :make ... ..	327
Заполнение списка с результатами на основе вывода Nodelint .....	328
Настройка makeprg и errorformat единственной командой .....	329

<b>Глава 18. Поиск в пределах проекта с помощью команд <code>grep</code>, <code>vimgrep</code> и других</b> .....	331
Рецепт 108. Вызов <code>grep</code> , не покидая Vim .....	331
Использование <code>grep</code> из командной строки .....	332
Вызов <code>grep</code> из редактора Vim .....	332
Рецепт 109. Настройка программы <code>grep</code> .....	333
Настройки по умолчанию команды <code>:grep</code> .....	333
Настройка <code>:grep</code> на вызов команды <code>ask</code> .....	334
Переход к строке и позиции в строке при использовании <code>ask</code> .....	336
Рецепт 110. Поиск с использованием внутреннего механизма поиска Vim .....	337
Поиск в файле с последующим поиском в проекте .....	337
История поиска и <code>:vimgrep</code> .....	338
<b>Глава 19. Набери X и пользуйся автодополнением</b> .....	339
Рецепт 111. Встречайте: автодополнение ключевых слов.....	339
Вызов функции автодополнения .....	341
Рецепт 112. Работа с меню функции автодополнения.....	341
Обзор списка слов без изменения документа .....	342
Изменение документа по мере прокручивания списка слов .....	343
Отклонение вариантов выбора .....	343
Фильтрация списка по мере ввода.....	343
Рецепт 113. Источники ключевых слов .....	344
Буфер .....	344
Подключаемые файлы .....	345
Индексные файлы.....	345
Объединяем все вместе .....	346
Рецепт 114. Автодополнение словами из словаря .....	347
Рецепт 115. Автодополнение целых строк .....	348
Рецепт 116. Автодополнение имен файлов.....	349
Рецепт 117. Контекстное автодополнение.....	351
<b>Глава 20. Поиск и исправление опечаток в Vim</b> .....	353
Рецепт 118. Проверьте свой текст .....	353
Принцип действия механизма проверки орфографии в Vim .....	354

Рецепт 119. Использование альтернативных орфографических словарей .....	355
Настройка диалекта языка .....	355
Получение словарей для других языков .....	356
Рецепт 120. Добавление слов в орфографический словарь...	356
Создание словаря для специальных терминов.....	357
Рецепт 121. Исправление орфографических ошибок в режиме вставки .....	358
Подготовка .....	358
Обычный способ: выход в командный режим .....	358
Быстрый способ: использовать орфографическое автодополнение .....	358
<b>Глава 21. Что дальше?</b> .....	360
21.1. Продолжайте практиковаться! .....	360
21.2. Настраивайте Vim под себя .....	360
21.3. Узнайте, как пользоваться пилой, и только потом точите ее .....	361
<b>Приложение 1. Настройка Vim в соответствии с личными предпочтениями</b> .....	363
A1.1. Изменяйте настройки на лету .....	363
A1.2. Сохраняйте настройки в файле vimrc .....	365
A1.3. Применение настроек для определенных типов файлов .....	367
<b>Предметный указатель</b> .....	369



## Глава 1. Путь Vim

Выполняемая нами работа содержит множество повторений. Будь то внесение одних и тех же небольших изменений в нескольких местах или перемещение между похожими фрагментами документа, мы повторно выполняем одни и те же действия. Все, что сможет упростить выполнение повторяющихся операций, поможет нам сэкономить наше время.

Редактор Vim оптимизирует такие повторяющиеся действия. Его эффективность во многом обусловлена возможностью отслеживать последовательность последних выполняемых операций. Мы всегда можем воспроизвести последние изменения нажатием одной клавиши. Это действительно очень мощная особенность, но она становится совершенно бесполезной для тех, кто не умеет фиксировать свои действия так, чтобы они выполняли полезную работу при повторном воспроизведении. Овладение данной концепцией является ключом к эффективному использованию Vim.

Команда «точка» станет нашей отправной точкой. Эта, казалось бы, простая команда – самый мощный инструмент в нашем арсенале и первый шаг на пути к овладению редактором Vim. Мы рассмотрим несколько простых задач редактирования, которые с помощью команды «точка» могут быть решены в мгновение ока. Хотя все задачи отличаются друг от друга, их решения во многом сходны. В каждом случае мы определим единую формулу редактирования, для выполнения которой требуется нажатие всего одной клавиши.

### Рецепт 1. Встречайте: команда «точка»

Команда «точка» позволяет повторять последнее изменение. Это самая мощная и гибкая команда в редакторе Vim.

Документация к редактору Vim просто отмечает, что команда «точка»: «повторяет последнее изменение» (см. :h .  <http://>

[vimdoc.sourceforge.net/html/doc/repeat.html#](http://vimdoc.sourceforge.net/html/doc/repeat.html#)). Вроде бы ничего особенного, но за этим простым определением скрывается мощная основа, делающая модель модального редактирования в Vim такой эффективной. Для начала выясним, что это за «последнее изменение».

Чтобы осознать всю мощь команды «точка», нужно понимать, что под «последним изменением» может скрываться все, что угодно. Изменением могут быть действия над отдельными символами, строками или даже над целым файлом.

Для демонстрации воспользуемся следующим фрагментом текста:

### [the\\_vim\\_way/0\\_mechanics.txt](#)

[http://media.pragprog.com/titles/dnvim/code/the\\_vim\\_way/0\\_mechanics.txt](http://media.pragprog.com/titles/dnvim/code/the_vim_way/0_mechanics.txt)

---

```
Line one
Line two
Line three
Line four
```

---

Команда **x** удаляет символ под курсором. Когда команда «точка» используется в этом контексте, под «последним изменением» Vim будет понимать удаление символа под курсором:

Нажатия клавиш	Содержимое буфера
{start}	Line one Line two Line three Line four
x	Line one Line two Line three Line four
.	Line one Line two Line three Line four
..	Line one Line two Line three Line four

Вернуть файл в исходное состояние можно, нажав клавишу **u** несколько раз, чтобы отменить изменения.

Команда **dd** тоже выполняет удаление, но она удаляет текущую строку целиком. Если использовать команду «точка» после команды

**dd**, тогда под «последним изменением» Vim будет понимать удаление текущей строки:

Нажатия клавиш	Содержимое буфера
{start}	Line one Line two Line three Line four
<b>dd</b>	Line one Line two Line three Line four
.	Line three Line four

Наконец, команда **>G** увеличивает отступ, начиная с текущей строки, до конца файла. После выполнения этой команды под «последним изменением» Vim будет понимать увеличение отступа от текущей строки до конца файла. Этот пример мы начали с того, что поместили курсор в начало второй строки, чтобы подчеркнуть отличия.

Нажатия клавиш	Содержимое буфера
{start}	Line one Line two Line three Line four
<b>&gt;G</b>	Line one Line two Line three Line four
<b>j</b>	Line one Line two Line three Line four
.	Line one Line two Line three Line four
<b>j.</b>	Line one Line two Line three Line four

Все команды – **x**, **dd** и **>** – выполняются в командном режиме, но изменения также создаются каждый раз при входе в режим вставки.

От момента входа в режим вставки (например, нажатием клавиши **i**) до выхода в командный режим (нажатием **<Esc>**) Vim записывает все нажатия клавиш. После создания такого изменения команда «точка» будет повторять эти нажатия (но прочитайте предупреждение во врезке «Перемещение по тексту в режиме вставки закрывает последнее изменение» в главе 2).

## Команда «точка» – микромакроопределение

Ниже, в главе 11 «Макросы», будет показано, как в редакторе Vim организовать запись последовательности из произвольного количества нажатий клавиш для повторного воспроизведения ее в будущем. Эта возможность позволяет сохранять повторяющиеся операции и воспроизводить их нажатием одной клавиши. Команду «точка» можно считать миниатюрным, или «микро-» (если хотите), макроопределением.

На протяжении этой главы будет показано несколько вариантов применения команды «точка». Кроме того, пара приемов работы с командой «точка» будет показана в рецептах 9 (глава 2) и 43 (глава 4).

## Рецепт 2. Не повторяйся

Для такой типичной операции, как добавление точки с запятой в конец строки, Vim предоставляет отдельную команду, объединяющую два шага в один.

Допустим, что у нас имеется следующий фрагмент кода на JavaScript:

### the\_vim\_way/2\_foo\_bar.js

[http://media.pragprog.com/titles/dnvim/code/the\\_vim\\_way/2\\_foo\\_bar.js](http://media.pragprog.com/titles/dnvim/code/the_vim_way/2_foo_bar.js)

```
var foo = 1
var bar = 'a'
var foobar = foo + bar
```

Нам требуется добавить точку с запятой в конец каждой строки. Для этого нужно переместить курсор в конец текущей строки и затем перейти в режим вставки, чтобы выполнить изменение. Команда **\$** выполнит такое перемещение, после чего мы сможем нажать последовательность **a**; **<Esc>**, чтобы выполнить изменение.

Чтобы выполнить задание полностью, нам могло бы потребоваться повторить ту же самую последовательность нажатий с остальными двумя строками, но это дело слишком уж нехитрое. Команда «точка» повторяет последнее изменение, поэтому мы можем просто выполнить `j$` дважды. Одно нажатие (`.`) экономит три (`a`; <Esc>). Экономия получается не очень большая, но эффективность приема будет расти с ростом количества повторений.

А теперь рассмотрим поближе этот шаблон: `j$.` Команда `j` перемещает курсор на одну строку вниз, а затем команда `$` перемещает его в конец строки. Нам пришлось нажать две клавиши, только чтобы вывести курсор в позицию, где можно применить команду «точка». Не кажется ли вам, что здесь есть что улучшить?

### ***Избавляйтесь от лишних перемещений***

Команда `a` переходит в режим вставки в позицию, находящуюся за позицией курсора. Однако есть команда `A`, выполняющая переход в режим вставки в позицию, находящуюся в конце текущей строки. Где бы ни находился курсор, нажатие на клавишу `A` выполнит переход в режим вставки и переместит курсор в конец строки. Иными словами, эта команда совмещает в себе последовательность команд `$a`. Во врезке «Две по цене одной» ниже мы узнаем, что в редакторе Vim имеется несколько подобных составных команд.

Ниже показано измененное решение предыдущей задачи:

Нажатия клавиш	Содержимое буфера
{start}	<code>var foo = 1 var bar = 'a' var foobar = foo + bar</code>
<code>A</code> ; <Esc>	<code>var foo = 1; var bar = 'a' var foobar = foo + bar</code>
<code>j</code>	<code>var foo = 1; var bar = 'a' var foobar = foo + bar</code>
<code>.</code>	<code>var foo = 1; var bar = 'a'; var foobar = foo + bar</code>
<code>j.</code>	<code>var foo = 1; var bar = 'a'; var foobar = foo + bar;</code>

Используя **A** вместо **\$a**, мы расширяем команду «точка». Вместо того чтобы перемещать курсор в *конец* строки, которую требуется изменить, мы просто перемещаем курсор в эту строку (в *любую позицию* в ней!). Теперь мы можем внести изменения в последовательность строк, просто нажимая **j**. столько раз, сколько требуется.

Одно нажатие – чтобы переместить курсор, и одно нажатие – чтобы выполнить изменение. Разве это не здорово?! Запомните этот прием, потому что мы встретимся с ним еще в нескольких примерах.

Несмотря на свою потрясающую простоту, эта формула не является универсальным решением. Представьте, что нам потребовалось добавить точку с запятой в пятьдесят строк, следующих друг за другом. В этом случае нажатие пары клавиш **j**. превращается в утомительный труд. Альтернативное решение этой задачи вы найдете в рецепте 30 в главе 5.

## Две по цене одной

Мы могли бы сказать, что команда **A** состоит из двух действий (**\$a**), совмещенных в одном нажатии клавиши. Однако она не является единственной командой такого рода. Многие команды Vim можно рассматривать как одноклавишные версии последовательностей из двух или более команд. В таблице ниже перечислено несколько таких команд. Сможете ли вы определить, что их объединяет?

Составная команда	Эквивалентная последовательность
<b>C</b>	<b>c\$</b>
<b>s</b>	<b>cl</b>
<b>S</b>	<b>^C</b>
<b>I</b>	<b>^i</b>
<b>A</b>	<b>\$a</b>
<b>o</b>	<b>A&lt;CR&gt;</b>
<b>O</b>	<b>ko</b>

Если вы вдруг обнаружите, что все время выполняете последовательность **ko** (или хуже того, **k\$a<CR>**), остановитесь! Подумайте о том, что вы делаете. Затем вспомните, что можно было бы использовать более простую команду **O**.

Сумеете ли вы заметить другую общую черту, объединяющую эти команды? Все они выполняют переход из командного режима в режим вставки. Поразмышляйте над этим и над тем, как это может влиять на команду «точка».

## Рецепт 3. Шаг назад, три вперед

Мы можем дополнить единственный символ пробелами (один слева, другой справа), используя идиоматическое для Vim решение. Сначала оно будет казаться странным, но это решение дает возможность повторения, что позволяет выполнить задание с минимумом усилий.

Представьте, что имеется такая строка кода:

### the\_vim\_way/3\_concat.js

[http://media.pragprog.com/titles/dnvim/code/the\\_vim\\_way/3\\_concat.js](http://media.pragprog.com/titles/dnvim/code/the_vim_way/3_concat.js)

```
var foo = "method("+argument1+", "+argument2+)";
```

Конкатенация строк в JavaScript никогда не выглядела особенно удобочитаемо, тем не менее мы могли бы повысить удобочитаемость, окружая каждый знак + пробелами, чтобы строка кода выглядела, как показано ниже:

```
var foo = "method(" + argument1 + ", " + argument2 + ")";
```

## Делайте изменения повторяемыми

Описываемая задача имеет следующее идиоматическое решение:

Нажатия клавиш	Содержимое буфера
{start}	var foo = "method("+argument1+", "+argument2+)";
f+	var foo = "method(" + argument1 + ", "+argument2+)";
s_+_<Esc>	var foo = "method(" + argument1 + ", "+argument2+)";
;	var foo = "method(" + argument1 + ", "+argument2+)";
.	var foo = "method(" + argument1 + ", "+argument2+)";
;	var foo = "method(" + argument1 + ", " + argument2 + ")";
;	var foo = "method(" + argument1 + ", " + argument2 + ")";

Команда S объединяет два шага в один: она удаляет символ под курсором и переходит в режим вставки. После удаления символа + мы вводим \_+ и покидаем режим вставки.

Один шаг назад и затем три вперед. Этот странный танец на клавиатуре сначала выглядит малопонятным, но он дает нам большой выигрыш: мы можем повторять изменения с помощью команды «точка»: все, что для этого нужно, — переместить курсор к следующему символу +, после чего команда «точка» повторит этот маленький танец.

## Делайте перемещения повторяемыми

В этом примере присутствует еще одна хитрость. Команда `f{char}` сообщает редактору Vim, что он должен найти следующее вхождение указанного символа и переместить курсор в его позицию (см. `:h f`  <http://vimdoc.sourceforge.net/html/doc/motion.html#f>). Поэтому, когда вводится последовательность `f+`, курсор перепрыгивает к следующему символу `+`. Подробнее о команде `f{char}` рассказывается в рецепте 49 в главе 8.

Выполнив первое изменение, мы могли бы перейти к следующему символу `+`, повторив команду `f+`, но есть более простой путь. Команда `;` повторяет последний поиск, произведенный с помощью команды `f`. Поэтому вместо команды `f+` мы можем использовать эту команду.

## Теперь все вместе

Команда `;` переносит курсор в позицию следующего символа `+`, а команда `.` повторяет последнее изменение, поэтому мы можем внести оставшиеся изменения, трижды введя последовательность `;`. Не кажется ли вам такой алгоритм действий знакомым?

Вместо того чтобы бороться с модальной моделью ввода в Vim, мы используем ее преимущества и убеждаемся, насколько она способна упростить решение поставленной задачи.

## Рецепт 4. Действие, повтор, возврат

Сталкиваясь с повторяющимися задачами, мы можем выработать еще более оптимальную стратегию редактирования, делая одновременно перемещения и изменения повторяющимися. В редакторе Vim имеется все необходимое для этого. Он запоминает наши действия и сохраняет их так, чтобы мы легко могли воспользоваться ими. В этом рецепте мы познакомимся с каждым из действий, которые Vim способен повторить, и узнаем, как отменить их.

Мы уже знаем, что команда «точка» повторяет последнее *изменение*. Поскольку существует огромное количество команд, которые могут интерпретироваться как изменения, команда «точка» оказывается чрезвычайно универсальной. Но некоторые команды могут повторяться другими средствами. Например, `@:` может повторить любую команду редактора Ex (как описывается в рецепте 31 в главе 5). Точно так же, нажав `&` (см. рецепт 92 в главе 14), можно повто-

рить последнюю команду подстановки `:substitute` (которая также является командой редактора Ex).

Если известно, как повторять действия, избегая необходимости вводить их каждый раз, мы можем добиться высочайшей эффективности. Сначала мы выполняем действие, затем повторяем его.

Но когда многого можно добиться лишь несколькими нажатиями клавиш, приходится быть внимательными. Слишком легко допустить ошибку. Многократное повторение `j.j.j` снова и снова начинает напоминать барабанную дробь. Что, если по ошибке мы нажмем клавишу `j` два раза подряд? Или хуже того, два раза нажмем клавишу `.`?

Всякий раз, когда редактор Vim упрощает повторение действия или перемещения, он всегда предоставляет способ отступить назад, если, войдя в раж, мы зайдем слишком далеко. В случае с командой «точка» мы всегда можем нажать клавишу `u`, чтобы отменить последнее изменение. Если лишний раз нажать клавишу `.` после использования команды `f{char}`, можно пропустить искомый символ. Но мы можем вернуться назад, нажав клавишу `;`, которая повторяет последнюю команду поиска `f{char}` в обратном направлении (см. рецепт 49 в главе 8).

Всегда полезно знать, как включается обратная передача, на случай, если по ошибке мы зайдем слишком далеко. В табл. 1 перечислены команды повторения, поддерживаемые редактором Vim, и соответствующие им обратные команды. В большинстве случаев можно использовать команду отмены. Неудивительно, что самая потерянная клавиша на моей клавиатуре – это клавиша `u`!

**Таблица 1. 1. Команды повторения и обратные им команды**

Цель	Команда	Повторение	Возврат
Выполнить изменение	<code>{edit}</code>	<code>.</code>	<code>u</code>
Найти следующий символ в строке	<code>f{char}/t{char}</code>	<code>;</code>	<code>;</code>
Найти предыдущий символ в строке	<code>F{char}/T{char}</code>	<code>;</code>	<code>;</code>
Найти следующее совпадение в документе	<code>/pattern&lt;CR&gt;</code>	<code>n</code>	<code>N</code>
Найти предыдущее совпадение в документе	<code>?pattern&lt;CR&gt;</code>	<code>n</code>	<code>N</code>
Выполнить подстановку	<code>:s/target/replacement</code>	<code>&amp;</code>	<code>u</code>
Выполнить последовательность изменений	<code>qx{changes}q</code>	<code>@x</code>	<code>u</code>

## Рецепт 5. Поиск и замена вручную

Для выполнения поиска с заменой в Vim имеется команда `:substitute`, но существует альтернативный прием, когда мы изменяем первое вхождение вручную, а затем отыскиваем и заменяем каждое последующее вхождение по одному. Команда «точка» могла бы спасти нас от переутомления, но существует другая, более привлекательная команда, осуществляющая переход между совпадениями.

В следующем фрагменте текста в каждой строке встречается слово «content»:

### [the\\_vim\\_way/1\\_copy\\_content.txt](http://media.pragprog.com/titles/dnvim/code/the_vim_way/1_copy_content.txt)

[http://media.pragprog.com/titles/dnvim/code/the\\_vim\\_way/1\\_copy\\_content.txt](http://media.pragprog.com/titles/dnvim/code/the_vim_way/1_copy_content.txt)

---

```
...We're waiting for content before the site can go live...
...If you are content with this, let's go ahead with it...
...We'll launch as soon as we have the content...
```

---

Допустим, что нам понадобилось заменить каждое слово «content» словом «сору». Вы можете подумать, что сделать это очень просто – достаточно лишь воспользоваться командой подстановки:

---

```
⇒ :%s/content/copy/g
```

---

Но минутку! В этом случае фраза «If you are content with this» («Если вас это устраивает») превратится во фразу «If you are 'copy' with this» («Если вас это копия»), лишенную смысла!

Мы столкнулись с проблемой, потому что слово «content» имеет два смысла. Один из них – синоним слова «сору» (копия), а другой – синоним слова «happy» (довольный). Технически мы имеем дело с омонимами (словами, разными по значению, но одинаковыми по написанию), но не это здесь главное. Главное здесь то, что вы должны следить за своими действиями.

Мы не можем слепо заменить каждое вхождение слова «content» словом «сору». Мы должны оценить каждое из найденных вхождений и ответить «да» или «нет» на вопрос о замене. Команда подстановки имеет свою область применения, и мы познакомимся с ней поближе в рецепте 89 в главе 14. А пока займемся альтернативным решением, соответствующим теме данной главы.

## Будьте экономны: выполняйте поиск без ввода лишних символов

Возможно, вы уже догадались, что команда «точка» – моя любимица. На втором месте у меня стоит команда **\***. Она выполняет поиск слова под курсором (см. :h \*  <http://vimdoc.sourceforge.net/htmldoc/pattern.html#star>).

Мы можем выполнить поиск слова «content», напечатав его в строке приглашения к вводу:

---

⇒ /content

---

или просто поместив курсор в требуемое слово и нажав клавишу **\***. Взгляните, как выглядит весь процесс:

Нажатия клавиш	Содержимое буфера
{start}	...We're waiting for content before the site can go live... ...If you are content with this, let's go ahead with it... ...We'll launch as soon as we have the content...
*	...We're waiting for content before the site can go live... ...If you are content with this, let's go ahead with it... ...We'll launch as soon as we have the content...
cwcopy<Esc>	...We're waiting for content before the site can go live... ...If you are content with this, let's go ahead with it... ...We'll launch as soon as we have the copy...
n	...We're waiting for content before the site can go live... ...If you are content with this, let's go ahead with it... ...We'll launch as soon as we have the content...
.	...We're waiting for copy before the site can go live... ...If you are content with this, let's go ahead with it... ...We'll launch as soon as we have the content...

Процесс начинается с установки курсора в слово «content», после чего вызывается команда **\*** поиска. Попробуйте сами выполнить это упражнение. В результате случится следующее: курсор перепрыгнет вперед, к следующему совпадению, и все найденные вхождения будут выделены цветом. Если вы не увидите подсвеченные совпадения, попробуйте выполнить команду :set hls и затем обратиться к рецепту 80 в главе 13 за более подробными разъяснениями.

Выполнив поиск слова «content», перейти к следующему совпадению можно простым нажатием клавиши **n**. В данном случае нажатие **\*nn** выполнит обход всех совпадений и вернет нас туда, откуда мы начали.

## **Делайте изменения повторяемыми**

После установки курсора в начало слова «content» мы можем изменить его. Для этого требуется выполнить два шага: удалить слово «content» и затем ввести слово замены. Команда **cw** удаляет символы от курсора до конца слова и выполняет переход в режим вставки, после чего можно ввести слово «сору». Vim записывает все нажатия клавиш, вплоть до выхода из режима вставки, поэтому вся последовательность **cwсору<Esc>** рассматривается как единое изменение. Нажатие клавиши **.** удалит символы до конца текущего слова и заменит его словом «сору».

## **Теперь все вместе**

Теперь все готово к действию! При каждом нажатии клавиши **h** курсор перемещается к следующему вхождению слова «content». А при нажатии клавиши **.** слово под курсором замещается словом «сору».

Если бы нам потребовалось слепо заменить все вхождения, мы могли бы просто отстучать на клавиатуре **h.n.n.** столько раз, сколько потребовалось бы для замены всех вхождений (хотя в этом случае можно было бы также использовать команду `:%s/content/copy/g`). Но нам нужно пропускать слова, не подлежащие замене. Поэтому после нажатия клавиши **h** мы проверяем смысл найденного совпадения и решаем, следует ли заменить его словом «сору». Если слово должно быть заменено, нажимаем клавишу **.** Если нет – не нажимаем. Независимо от принятого решения, чтобы перейти к следующему вхождению, снова нажимаем клавишу **h**. И так, пока не будет выполнена замена всех требуемых вхождений.

## **Рецепт 6. Формула точки**

Мы рассмотрели три простые задачи редактирования. Несмотря на различия между ними, мы смогли решить их с помощью команды «точка». В данном рецепте мы сравним найденные решения и выявим общий шаблон – оптимальную стратегию редактирования, которую я назвал «формула точки».

## **Обзор решений трех задач редактирования с помощью команды «точка»**

В рецепте 2 нам потребовалось добавить точку с запятой в конец каждой строки. Мы изменили первую строку нажатием клавиш **A**; <Esc>, благодаря чему появилась возможность повторить изменение в каждой последующей строке с помощью команды «точка». Мы могли бы перемещаться между строками, выполняя команду **j**, и выполнить оставшиеся изменения, просто последовательно нажимая **j**. столько раз, сколько потребуется.

В рецепте 3 нам понадобилось окружить каждый символ + пробелами с обеих сторон. Чтобы найти этот символ в тексте, мы воспользовались командой **f+** и затем задействовали команду **s**, чтобы заменить один символ тремя. Благодаря такому подходу мы получили возможность выполнить поставленную задачу, нажав клавиши **;**. несколько раз.

В рецепте 5 нам нужно было заменить каждое вхождение слова «content» словом «сору». Для инициации поиска требуемого слова мы использовали команду **\*** и затем выполнили команду **cw** для изменения первого вхождения. После этого мы получили возможность использовать команду **h** для перехода к следующему вхождению и команду **.** для применения того же изменения. Мы могли бы выполнить поставленную задачу, просто нажимая клавиши **h**. столько раз, сколько потребуется.

### **Идеальное решение: одна клавиша для перехода и одна для изменения**

Во всех примерах мы использовали команду «точка», чтобы повторить последнее изменение. Но это не единственная общая черта. Для перемещения к следующей позиции применения изменений также потребовалось единственное нажатие клавиши.

Мы использовали одно нажатие для перемещения курсора и одно нажатие для выполнения изменения. Разве можно придумать что-то еще более удобное? Это – идеальное решение. Мы еще не раз столкнемся с этой стратегией редактирования, поэтому для краткости будем называть ее «формула точки».



## **Часть I. РЕЖИМЫ**

Редактор Vim имеет модальный пользовательский интерфейс. То есть результат нажатия любой клавиши зависит от режима, активного в данный момент. Очень важно знать, какой режим активен в данный момент и как переключать режимы Vim. В этой части книги вы узнаете, как действует каждый режим и для каких целей они используются.