第 1 章 vim 解决问题的方式

技巧 1 结识 .命令

- .命令可以重复上次的修改
  - >g 会增加从当前行到文档末尾处的缩进层级
  - 每次进入插入模式也会形成一次修改。从进入插入模式的那一刻起，直到返回普通模式为止，vim 会记录每一个按键操作。
  - .命令是一个微型的宏

技巧 2 不要自我重复

- a 命令在当前光标之后添加内容

- a 命令在当前行的结尾添加内容

技巧 3 以退为进

- f{char}命令让 vim 查找下一处指定字符出现的位置，如果找到了，就直接把光标移动到那里
- ;命令会重复上次 f 命令所查找的字符
- s 命令先删除光标下的字符，然后进入插入模式

技巧 4 执行、重复、回退

- @:可以重复任意 ex 命令
- &重复上次的:substitute 命令

目的	操作	重复	回退
做出一个修改	{edit}	.	u
在行内查找下一指定字符	f{char}/t{char}	;	,
在行内查找上一指定字符	f{char}/t{char}	;	,
在文档中查找下一处匹配项	/pattern<cr>	n	n
在文档中查找上一处匹配项	?pattern<cr>	n	n
执行替换	:s/target/replacement	&	u
执行一系列修改	qx{changes}q	@x	u

技巧五 查找并手动替换

- *命令可以查找当前光标下的单词

技巧六 结识.范式

第 2 章 普通模式

技巧 7 停顿时请移开画笔

技巧 8 把撤销单元切成块

技巧 9 构造可重复的修改

- 如果在插入模式中使用了<up>、<down>、<left>、<right>这些光标键，将会产生一个新的撤销块。会对.命令的操作产生影响
- 使用更为精准的 aw 文本对象

技巧 10 用次数做简单的算术运算

- <c-a>和<c-x>命令对数字执行加和减操作。在不带数字执行时，它们会逐个加减，但如果带一个次数前缀，那么就可以用它们加减任意整数。
  - 如果光标不在数字上，会把当前光标之上或之后的数值加上[count]
  - vim 会把 0 开头的数字解释为八进制值，可以加入 set nrformats=将所有数字设为十进制

技巧 11 能够重复，就别用次数

技巧 12 双剑合璧，天下无敌

命令	用途
c	修改
d	删除
y	复制到寄存器
g~	反转大小写 （可视化模式下）
gu	转为小写
gu	转为大写
>	增加缩进
<	减小缩进
=	自动缩进
!	使用外部程序过滤{motion}所跨越的行

- >>缩进当前行
- guap 把整段文字转换为大写
- vim 的语法只有一条额外的规则，即当一个操作符命令被连续调用两次时，它会作用于当前行
  - gUgU 或 gUU 可以作用于当前行

第 3 章 插入模式

技巧 13 在插入模式中可即时更正错误

- <C-h>删除前一个字符（同退格键）
- <C-w>删除前一个单词
- <C-u>删至行首

技巧 14 返回普通模式

按键操作	用途
Esc>	切换普通模式
<C-[	切换到普通模式
<C-o>	切换到插入-普通模式

- 插入-普通模式 在此模式中，可以先执行一个普通模式，执行完后，马上回到插入模式
- zz 让当前行显示在窗口正中；<C-o>zz 在插入-普通模式中触发这条命名让正中显示后直接 j 回到插入模式

技巧 15 不离开插入模式，粘贴寄存器中的文本

- <C-r>{register}插入寄存器中的内容，但是当 textwidth 或 autoindent 选项被激活了的话，那么最终会出现不必要的换行或额外的缩进
- <C-r><C-p>{register}会更智能些，它会按照原意插入寄存器中的文本，并修正任何不必要的缩进

技巧 16 随时随地做运算

- 表达式寄存器允许我们做一些运算，并把运算结果直接插入到文档中
- 在插入模式中，输入 <C-r>=就可以访问这一寄存器。这条命令会在屏幕的下方显示一个提示符，我们可以在其后输入要执行的表达式。输入表达式后敲一下<CR>，Vim 就会把执行的结果插入到文档的当前位置了。

技巧 17 用字符编码插入非常用字符

- 在插入模式下输入<C-v>{code}其中{code}是要插入字符的编码
- 把光标移动到字符上面并按 ga 命令，下方就会列出编码信息（十进制和十六进制）

按键操作	用途
<C-v>{123}	以十进制字符编码插入字符
<C-v>u{1234}	以十六进制字符 c 编码插入字符
<C-v>{nondigit}	按原义插入非数字字符
<C-j>{char1}{char2}	插入以二合字母{char1}{char2}表示的字符

技巧 18 用二合字母插入非常用字符

技巧 19 用替换模式替换已有文本

- 虚拟替换模式 gR

第 4 章 可视化模式

技巧 20 深入理解可视化模式

- viw
- <C-g>可以在可视化模式及选择模式间切换，在选择模式中输入任意可见字符，此字符会替换所选内容并切换到插入模式

技巧 21 选择高亮选区

命令	用途
v	激»面向字符的可视模式
V	激»面向行的可视模式
<C-v>	激»面向列块的可视模式
gv	重选上次的高亮选区
o	切换高亮选区的活动端

技巧 22 重复执行面向行的可视命令

- 当使用 . 命令重复对高亮选区所做的修改时，此修改会重复作用于相同范围的文本(验证失败)

技巧 23 只要可能，最好用操作符命令，而不是可视命令

- 可视化模式下可以使用 U 命令来大写所选字符
- vit 选择标签内文本
- 当一条可视模式命令被重复执行时，它会影响相同数量的文本
- 在这个模式下 . 命令有时会有一些异常的表现。

技巧 24 面向列块的可视模式编辑表格数据

技巧 25 修改列文本

- 列块可视化模式中，删除操作会影响所有被选中的，但插入操作只影响顶行，而且 a i o 等命令无效
    - 因为在可视化模式下，a i 被当作一个文本对象的组成部分，而 o 有其他的作用

技巧 26 在长短不一的高亮块后添加文本

- I A 命令

第 5 章 命令行模式

技巧 27 结识 Vim 的命令行模式

- 有些命令在插入模式和命令行模式中可以通用。例如，可以用 <C-w> 和 <C-u> 分别删除至上个单词的开头及行首，也可以用 <C-v> 或 <C-k> 来插入键盘上找不到的字符，还可以用 <C-r>{register} 命令把任意寄存器的内容插入到命令行

技巧 28 在一行或多个连续行上执行命令

- 用行号作为地址
  - 如果只输入一条只包含数字的 Ex 命令，那么 Vim 就把这个数字解析成一个地址，并把光标移动到该数字所指定的行上
    - :1 跳转到第一行，:$跳转到最后一行
  - :p 或者:print 在命令行输出当前行内容
    - :3p 光标移动到第三行，然后显示该行内容
    - :3d 光标移动到第三行，并删除此行
- 用地址指定一个范围
  - :2,5p 打印从第二行到第五行之间的每一行内容（包含第二和第五行），运行完后，光标停留在第五行
  - .代表当前行
  - %代表文件中所有的行
- 用高亮选区指定范围
  - 可视化模式下按下:键，命令行上会预先填充一个范围 :'<,'>，然后输入一条 Ex 命令，使它作用在每个被选中的行上
  - '<是代表高亮选区首行的位置标记
  - '>是代表高亮选区的最后一行
- 用模式指定范围
  - :/<html>/,/\/html>/p 也符合 :{start},{end}
    - {start} 是 /<html>/，而{end} 地址是 /</html>/

- 用偏移对地址进行修正
  - :{address}+n 如果 n 被省略，缺省值为 1
    - :.,.+3p 等于 当前行到下三行

技巧 29 使用 :t 和 :m 命令复制和移动

- :copy 及其 :t 可以把一行或者多行从文档的一部分复制到另一部分
  - :{range}t{address}
- :move 可以把一行或多行文档移到文档其他地方

技巧 30 在指定范围上执行普通模式命令

- :%normal A; 会在全文每行末尾添加分号

技巧 31 重复上次的 EX 命令

- . 命令可以重复上次的普通模式命令。然而，如果想重复上次的 Ex 命令的话，我们得使用 @: 才行。
- : 寄存器总是保存着最后执行的命令行命令

技巧 32 自动补全 Ex 命令

技巧 33 把当前单词插入命令行

- 在 Vim 的命令行下，<C-r><C-w>映射项会复制光标下的单词并把它插入到命令行中。
- *命令等效于输入 /\<<C-r><C-w>\><CR>

技巧 34 回溯历史命令

- <UP> <Down>

命令	动作
q/	打开查找命令历史的命令行窗口
q:	打开 Ex 命令历史的命令行窗口
<C-f>	从命令行模式切换到命令行窗口

技巧 35 运行 shell 命令

- !叹号前缀
- <C-z>挂起 Vim 所属的进程，并把控制权交还给 bash
- jobs 查看当前作业列表

- fg 命令唤醒一个被挂起的作业

第 6 章 管理多个文件

技巧 36 用缓冲区列表管理打开的文件

- :ls 列出所有被载入内存中的缓冲区的列表
- bnext
- % 符号指明哪个缓冲区在当前窗口中可见，而 # 符号则代表轮换文件。按<C-^>可以在当前文件和轮换文件间快速切换
- :bprev 和 :bnext 在列表中反向或正向移动，每次移动一项；而 :bfirst 和 :blast 则分别跳到列表的开头和结尾
- 用:buffer N 命令直接凭编号跳转到一个缓冲区
- :bufdo 命令允许我们在 :ls 列出的所有缓冲区上执行 Ex 命令
  - :argdo 更加实用
- 删除缓冲区，可以用 :bdelete 命令
  - :bdelete N1 N2 N3
  - :N,M bdelete

技巧 37 用参数列表将缓冲区分组

- 用 Glob 模式指定文件
- :args commnads Vim 会在 shell 中执行反撇号括起来的命令，然后将结果输出作为:args 的参数

技巧 38 管理缓冲区列表

- 修改但未保存的文件会有 + 号

技巧 39 将工作区切分成窗口

- 用 s 命令可以水平切分此窗口，使之成为两个高度相同的窗口；或者可以用 v 命令对其进行垂直切分，这样会产生两个宽度相同的窗口。
- w h j k l
- close 关闭活动窗口
- on 只保留活动窗口，关闭其他所有窗口

技巧 40 用标签页将窗口分组

- :lcd {path} 命令让我们可以设置当前窗口的本地工作目录
  - :lcd 只影响当前窗口，而非当前标签页
- 用 :windo lcd {path} 命令为所有这些窗口设置本地工作目录。

第 7 章 打开及保存文件

技巧 41 用 :edit 命令打开文件

- `:pwd` 打印工作目录
- `:edit %<Tab>` %符号代表缓冲区的完整文件路径
  - `:edit %:h<Tab>` :h 修饰符会去除文件名
- cnoremap %% getcmdtype( ) == ':' ? expand('%:h').'/' : '%%'
  - 当你在 Vim 的命令行提示符后输入 %% 时，它就会被自动展开为活动缓冲区所在目录的路径

技巧 42 使用 :find 打开文件

- 需要预设 path
  - set path+=路径
  - /** 匹配该目录下所有子目录

技巧 43 使用 netrw 管理文件系统

- `:edit .` 打开文件管理器，并显示当前工作目录
- `:Explore` 打开文件管理器，并显示活动缓冲区所在目录
- 除:Explore 外，netrw 还提供了:Sexplore 及 :Vexplore 命令，这两条命令分别在一个水平切分窗口及垂直切分窗口里打开文件管理器。

技巧 44 把文件保存到不存在的目录中

- `:!mkdir -p %:h` -p 参数 创建任何不存在的中间目录

技巧 45 以超级用户权限保存文件

- `:w !sudo tee % > /dev/null`
  - `:write !{cmd}` 命令会把缓冲区的内容作为标准输入传给指定的{cmd}

第 8 章 用动作命令在文档中移动

技巧 46 让手指保持在本位行

- h j k l

技巧 47 区分实际行与屏幕行

- gj gk 是按照屏幕行向下及向上移动
  - 当一行显示不下，会有多个屏幕行的时候，可以在屏幕行之间移动

命令	光标动作
j	向下移动一个实际行
gj	向下移动一个屏幕行
k	向上移动一个实际行
gk	向上移动一个屏幕行
0	移动到实际行的行首
g0	移动到屏幕行的行首
^	移动到实际行的第一个非空白字符
g^	移动到屏幕行的第一个空白字符
$	移动到实际行的行尾
g$	移动到屏幕行的行尾

技巧 48 基于单词的移动

命令	光标动作
w	正向移动到下一单词的开头
g	反向移动到当前单词/上一单词的开头
e	正向移动到当前单词/下一单词的结尾
ge	反向移动到上一单词的结尾

- 一个单词由字母、下划线、数字，或其他非空白字符的序列组成，单词间以空白字符为间隔
- 一个字串由空白字符序列组成，字串间以空白字符分隔

技巧 49 对字符进行查找

- f{char} 在光标位置与当前行尾政治家查找指定字符
  - Vim 会记录上次执行过的 f{char} 命令，随后用 ; 命令就可以重复该命令了
  - 用 , 命令就可以再跳回来

命令	用途
f{char}	正向移动到下一个{char}所在之处
F{char}	反向移动到上一个{char}所在之处
t{char}	正向移动到下一个{char}所在之处的前一个字符上
T{char}	反向移动到上一个{char}所在之处的前一个字符上

技巧 50 通过查找进行移动

- 查找一个开动作

技巧 51 用精确的文本对象选择区

- `a) 或 abi) ib`
- `a} 或 aB i} iB`
- `a] i]`
- `a> i>`
- `a' i'`
- `a" i"`
- `ai`
- at it

技巧 52 删除周边，修改内部

文本对象	选择范围
iw	当前单词
aw	当前单词及一个空格
iW	当前字串
aW	当前字串 及一个空格
is	当前句子
as	当前句子 及一个空格
ip	当前段落
ap	当前段落 及一个空格

技巧 53 设置位置标记，以便快速跳回

位置标记	跳转到
两个斜撇	当前文件中上次跳转动作之前的位置
斜撇 .	上次修改的地方

斜撇 ^　　上次插入的地方
斜撇 [　　上次修改或复制的起始位置
斜撇]　　上次修改或复制的结束位置
斜撇 <　　上次高亮选区的起始位置
斜撇 >　　上次高亮选区的结束位置

技巧 54 在匹配括号间跳转

- % 命令允许我们在一组开闭括号间跳转

第 9 章 在文件间跳转

技巧 55 遍历跳转列表

- 命令允许对跳转列表进行遍历
- :jumps 查看跳转列表

命令	用途
[count]G	跳转到指定的行号
/patten/?pattern/n/N	跳转到下一个/上一个模式出现之处
%	跳转到匹配的括号所在之处
(/)	跳转到上一句/下一句的开头
{/}	跳转到上一段/下一段的开头
H/M/L	跳转到屏幕最上方/正中间/最下方
gf	跳转到光标下的文件名
<C-]>	跳转到光标下的关键字的定义之处
'/斜撇	跳转到一个位置标记

- 在插入模式中按一下，你会发现这和按 键的效果是一样的，因为 Vim 本来就把 和 当成同一个东西。

技巧 56 遍历改变列表

技巧 57 跳转到光标下的文件

技巧 58 用全局位置标记在文件间快速跳转

第 10 章 复制与粘贴

技巧 59 用无名寄存器实现删除、复制与粘贴操作

技巧 60 深入理解 Vim 寄存器

- "{register} 指定要使用的寄存器
- delete yank put
- 我们怎样才能删除文本而不把其内容复制到任何寄存器？
  - 使用名为“黑洞”的特殊寄存器
  - "_d{motion} 会执行真正的删除操作
- 无名寄存器
  - 倘若我们没有指定要使用的寄存器，Vim 将缺省使用无名寄存器
  - " "使用两个双引号
  - x、s、d{motion}、c{motion} 与 y{motion}命令（以及它们对应的大写命令）都会覆盖无名寄存器中的内容。无论哪一种情况，都可以通过加"{register}前缀来指定另外一个寄存器，但无名寄存器总是缺省的
- 复制专用寄存器（"0）
  - 当我们使用 y{motion}命令时，要复制的文本不仅会被拷贝到无名寄存器中，而且也被拷贝到了复制专用寄存器中
- 有名寄存器（“a –”z）
- 系统剪贴板（“+）与选择专用寄存器（”*）
- 表达式寄存器 ("=)
  - 当我们从表达式寄存器获取内容时，Vim 将跳到命令行模式，并显示提示符“=”。这时，我们可以输入一段 Vim 脚本表达式并按执行，如果返回的是字符串（或者可被强制转换成字符串的数据），Vim 将会使用它。

寄存器	内容
"%	当前文件名
"#	轮换文件名
".	上次插入的文本
":	上次查找的 Ex 命令
"/	上次查找的模式

技巧 61 用寄存器中的内容替换高亮选区的文本

技巧 62 把寄存器的内容粘贴出来

- p 命令把寄存器中的文本粘贴到光标之后
- P 命令把寄存器中的文本粘贴到光标之前

技巧 63 与系统剪贴板进行交互

第 11 章 宏

技巧 64 宏的读取与执行

- q 键既是“录制”按钮，也是“停止”按钮
  - q {register} 开始录制，直到再次按下 q 键为止
- 用@{register}命令执行指定寄存器的内容
  - 我们在第一行用 @a 回放宏，而在下一行用 @@ 来回放同样的宏。

技巧 65 规范光标位置、直达目标以及中止宏

技巧 66 加次数回放宏

技巧 67 在连续的文本行上重复修改

- ~ 将某单词第一个字母大写

技巧 68 给宏追加命令

- 在我们输入 qa 时，Vim 将开始录制接下来的按键操作，并将它们保存到寄存器 a 中，这会覆盖该寄存器原有的内容。如果我们输入的是 qA 的话，Vim 也会录制按键操作，但会把它们附加到寄存器 a 原有的内容之后。

技巧 69 在一组文件中执行宏

- 以并行方式执行此宏

- 请仔细想一想，运行 :argdo normal @a 将对参数列表内的所有缓冲区执行我们刚录制的宏，当然也包括那个在录制宏时被修改的文件。因此，第一个缓冲区的内容将被两次封装于同一个模块中。
- 为了避免此类问题，我们将执行 :edit! ，放弃针对第一个缓冲区所做的所有修改
- 以串行方式执行宏

技巧 70 用迭代求值的方式给列表编号

技巧 71 编辑宏的内容

第 12 章 按模式匹配及按原义匹配

技巧 72 调整查找模式的大小写敏感性

- 通过使用元字符 ，可以覆盖 Vim 缺省的大小写敏感性设置。

技巧 73 按正则表达式查找时，使用

- 我们可以利用 。该元字符将会激活 verymagic 搜索模式，即假定除_、大小写字母以及数字 0 到 9 之外的所有字符都具有特殊含义

技巧 74 按原义查找文本时，使用

- 在正则表达式中使用的特殊字符，在按模式查找时用起来很顺手，但如果我们想按原义查找文本时，它们就变成了阻碍。使用 very nomagic 原义开关，可以消除附加在 .、* 以及 ? 等大多数字符上的特殊含义。

技巧 75 使用圆括号捕获子匹配

技巧 76 界定单词的边界

- 在 very magic 搜索模式下，用 符号表示单词定界符
- 我们可以在圆括号前面加上%，指示 Vim 不要将括号内的内容赋给寄存器 \1

技巧 77 界定匹配的边界

- 有时候，我们可能想指定一个范围较广的模式，但只对匹配结果的一部分感兴趣。Vim 中的元字符。
- 环视表达式
  - 从功能上讲，元字符 。

技巧 78 转义问题字符

- 当进行正向查找时，我们必须要转义符号 /。而且无论执行的是 very magic 查找（使用模式开关 还是 verynomagic 查找（使用原义开关），都需要转义
- 反向查找时要转义?号

第 13 章 查找

技巧 79 结识查找命令

- 指定查找的方向
  - 当我们使用 / 键执行一次查找时，Vim 将进行正向扫描。而如果是用 ? 键调出查找提示符的话，Vim 则会进行反向查找
- 重复上一次查找
  - n 命令用于跳转到下一处匹配，而 N 命令则用于跳转到上一处匹配
  - 如果我们用 / 执行一次正向查找，n 将继续向下查找；而如果最初的查找命令是 ?的话，n 将继续向上查找

技巧 80 高亮查找匹配

技巧 81 在执行查找前预览第一处匹配

- incsearch

技巧 82 统计当前模式的匹配个数

- 虽然没有任何方法可以让查找命令统计当前文档中的匹配个数，但是用下面这条命令就可以做到这一点：
  - :%s///gn
  - 实际上，我们调用的是:substitute 命令，但标志位 n 会抑制正常的替换动作。该命令不会对每处匹配进行替换，而是简单地统计匹配的次数，并将结果显示到命令行上。此处我们将查找域留空，旨在让 Vim 使用当前的查找模式。替换域（由于标志位 n 的缘故）不管怎样都将会被忽略，因此也可以将其留空。

技巧 83 将光标偏移到查找匹配的结尾

- /e
  - 我们使用 /lang/e 进行查找，该命令会像我们期望的那样，将光标置于查找匹配的结尾

- 当我们把查找域留空时，Vim 将重用上一次的查找模式，因此，该命令将使用偏移重复上一次查找。

技巧 84 对完整的查找匹配进行操作

- 我们将 //e 作为一个动作命令使用，其范围涵盖由查找匹配的起始到结尾之间的全部内容。

技巧 85 利用查找历史，迭代完成复杂的模式

- 用 q/ 调出命令行窗口。此窗口与一个常规的 Vim 缓冲区差不多，不过它的内容是查找历史，每行显示一条（参见结识命令行窗口）

技巧 86 查找当前高亮选区中的文本

- 在可视模式下，* 命令将查找光标下的单词

第 14 章 替换

技巧 87 结识 substitute 命令

- :[range]s[ubstitute]/{pattern}/{string}/[flags]
- 标志位
  - h s_flags
  - 标志位 g 使得 subsititute 命令可在全局范围内执行
  - 标志位 c 让我们有机会可以确认或拒绝每一处修改
  - 标志位 n 会抑制正常的替换行为
- 替换域中的特殊字符

符号	描述
插入一个换行符	
插入一个制表符	
\	插入一个反斜杠
\1	插入第一个子匹配
\2	插入第二个子匹配（最多到\9）
\0	插入匹配模式的所有内容
&	插入匹配模式的所有内容
~	使用上一次调用:substitute 时的{string}
={Vim script}	执行{Vim script} 表达式，并讲返回的结果作为替换

技巧 88 在文件范围内查找并替换每一处匹配

- 在缺省情况下，substitute 命令仅仅作用于当前行，而且只会修改第一处匹配。因此，为了在整个文件的范围内修改每一处匹配，我们必须指定范围，并使用标志位 g
  - g 看似为全局之意（global）,但实际上，它仅表示“当前一整行范围”

技巧 89 手动控制每一次替换操作

- 我们可以用标志位 c 控制:substitute 命令的行为。

技巧 90 重用上次的查找模式

技巧 91 用寄存器的内容替换

- 通过输入 {register}，我们可以将寄存器的内容插入到命令行。 search/tag-heirarchy.rb

技巧 92 重复上一次 substitute 命令

- :%s//~/&
  - 这条命令可以详解为如下指令：用同样的标志位、同样的替换字符串、同样的查找模式以及新的执行范围 %，重复上一次 substitute 命令。换句话说，该命令表示在整个文件的范围内重复上一次替换操作。
  - g&

技巧 93 使用子匹配重排 CSV 文件的字段

技巧 94 在替换过程中执行算术运算

- 在 Vim 中，通过调用函数 submatch(0)，即可得到当前匹配的内容

技巧 95 交换两个或更多的单词

- 至于替换的内容，我们必须通过执行一小段 Vim 脚本才能获得。这意味着要在替换域中使用符号 =
- 采用字典的设定

技巧 96 在多个文件中执行查找与替换

- 元字符
- :argdo 命令，它允许我们在一组文件中运行 Ex 命令
  - 但是首先一点，我们必须通过以下命令，把所有工程文件加到参数列表中：⇒:args **/*.txt

第 15 章 global 命令

技巧 97 结识 global 命令

- :global 命令通常采用以下形式
  - :[range] global[!] /{pattern}/ [cmd]
  - 在缺省情况下，:global 命令的作用范围是整个文件（%）
  - 如果我们不指定任何 [cmd]，Vim 将缺省使用 :print。

技巧 98 删除所有包含模式的文本行

- :vglobal 或简写的 :v 命令，恰好与 :g 命令的操作相反。也就是说，它用于在指定模式的非匹配行上执行 Ex 命令。

技巧 99 将 TODO 项收集至寄存器

- 通过把 :global 和 :yank 这两条命令结合在一起，我们可以把所有匹配{pattern}的文本行收集到某个寄存器中。
- 要用大写字母 A 引用寄存器。这意味着 Vim 将把内容附加到指定的寄存器，而用小写字母 a 的话，则会覆盖原有寄存器的内容。

技巧 100 将 CSS 文件中所有规则的属性按照字母排序

- :g/{pattern}/[range][cmd]
  - 用:g/{pattern} 匹配作为参考点，动态设置 [cmd] 的[range]。
- :global 命令的广义形式如下所示：
  - :g/{start}/ .,{finish} [cmd]