VIM命令手册Vim(Vi IMproved)

命令

[1.打开文件操作 1](#_Toc309410857)

[2.存储及退出文件 2](#_Toc309410858)

[3.模式转换 2](#_Toc309410859)

[4.光标移动 3](#_Toc309410860)

[5.插入 4](#_Toc309410861)

[6.删除 4](#_Toc309410862)

[7.复制和黏贴 5](#_Toc309410863)

[8.查找 5](#_Toc309410864)

[9.替换 6](#_Toc309410865)

[10.编辑多个文件 6](#_Toc309410866)

[11.其他命令 6](#_Toc309410867)

[12分屏操作 7](#_Toc309410868)

[13 vim的帮助指令 9](#_Toc309410883)

#### ****1.打开文件操作****

|  |  |
| --- | --- |
| vi | 进入vi而不读入任何文件 |
| vi filename | 进入vi并读入指定名称的文件 |
| vi filename....filename | 打开多个文件，依次进行编辑 |
| vi +n filename | 进入vi并且由文件的第几行开始。 |
| vi + filename | 进入vi并且由文件的最后一行开始。 |
| vi +/pattern filename | 打开文件，并将光标置于第一个与pattern匹配的串处 |
| vi -r filename | 在上次正用vi编辑时发生系统崩溃，恢复filename |
| vim -R file | 只读方式打开文件 |
| vimdiff fileA fileB | 比较编辑两个不同的文件 |

#### ****2.存储及退出文件****

|  |  |
| --- | --- |
| 保存 | |
| :wq 或者 :x | 存文件，并且退出vi |
| ZZ | 保存并退出 (:x 表示仅在需要时保存，ZZ不需要输入冒号并回车) ZZ = wq |
| :w | 保存当前文件 |
| :w filename | 写入指定文件，相当于另存为,但未退出vi（若未指定文件名则为当前工作的文件名）  可带行范围 :n1,n2 w filename |
| :w /tmp/1 | 既然没法存盘，不想放弃所做的所有修改，先临时存到/tmp/1 |
| :saveas <path/to/file> | 另存为 <path/to/file> |
| 退出 | |
| :q | 不作任何修改并退出vi (多屏时用qa) |
| :q! | 放弃任何修改并退出vi |
| ZQ | 无条件退出 |
| 编辑 | |
| :e filename | 打开文件filename进行编辑[相当于退出当前的并打开另一个文件] |
| :e! | 放弃修改文件内容，重新载入该文件编辑 |
| :e | 重新载入【类似刷新】 |
| :w !sudo tee % | 以普通用户身份打开的vim ，修改后以root身份保存 |
| 其他 | |
| ctrl + z / fg | 暂时挂到后台/ 跳回编辑页面 |
| :f 或 Ctrl-g | 显示文件名,当前光标所在行的行号,总的行数,以及当前行所在文件中的百分比和当前光标所在的列的信息 |
| :!command | 暂时退出vi并执行shell指令，执行完毕后再回到vi |
| :r!command | 将命令command的输出结果放到当前行【强大】 |
| :sh | 暂时退出vi到系统下，结束时按Ctrl + d则回到vi。 |
| vim -x exam.txt | 新编辑的文件加密，会提示输入密码 |
| :X | 文件加密，会提示输入密码 |
| :map | 列出当前已定义的映射  (如何定义映射？) |
| :jumps | 列出关于你曾经跳转过的位置的列表，你最后一个跳转的位置被特别以一个”>”号标记 |
| :marks | 列出自定义的标记，包含VIM内部自定义的标识 |
| :%!nl | 要对包含空行的所有行进行编号(所有行之前插入行号^I) |
| :version | 显示VIM版本信息，包含特性、编译方式及配置文件目录 |
| :n1,n2 w ! command | 将文件中n1行至n2行的内容作为command的输入并执行之，若不指定n1，n2，则表示将整个文件内容作为command的输入【注意空格】例如  :1,4 w! grep tom |
| :n1,n2 w file |  |
| :options | 打开一个新窗口, 在该窗口的最开头的注释下面是一个选项列表(do what?) |
| :r filename | 在光标所在处插入一个文件的内容(read)  可加入数字 :nr filename |
| :f filename | 改变编辑中的文件名(file)  相当于复制了一个文件，执行这个命令后，新编辑不会应用于原文件 |
| gf | vim特殊打开文件的方法，打开光标所在处的 word 为名的文件，当然，这个文件要在当前目录内，否则会创建新文件 |

#### ****3.模式转换****

1) **命令行模式(command mode）**

控制屏幕光标的移动，字符、字或行的删除，移动复制某区段及进入Insert mode下，或者到 last line mode

a i o 进入插入模式

：进入底行模式

    2) **插入模式（Insert mode）** <i/a/o/O> Insert 2Insert

只有在Insert mode下，才可以做文字输入

按「ESC」键可回到命令行模式。

    3) **底行模式（last line mode）** Normal模式

将文件保存或退出vi，也可以设置编辑环境，如寻找字符串、列出行号……等，使用命令执行编辑操作等

可缩写(:set 🡪:se) 补齐(tab键的功效+CTRL-D) 历史记录(借助CTRL-UP)

/ (查找) ?(反查)

1. **Visual模式**<v / V / CTRL-v /r>

#### ****4.光标移动****

|  |  |
| --- | --- |
| 字符 默认1，可加n | |
| h 或← | 光标左移一个字符[回退键Backspace] |
| l 或→ | 光标右移一个字符[空格键Space] |
| j 或↓ | 向下移动一行 |
| k 或↑ | 向上移动一行 |

BP:

hkjl作为移动光标【important，同方向键，建议使用这四个】

|  |  |
| --- | --- |
| 单词 默认1，可加n | |
| w | 光标跳到下个word的第一个字母 [常用] |
| W | 移到下一个字的开头，忽略标点符号 |
| b | 光标回到上个word的第一个字母 |
| B | 移到前一个字的开头，忽略标点符号 BACK |
| e | 光标跳到下个word的最后一个字母 |
| E | 移到下一个字的结尾，忽略标点符号 END |

|  |  |
| --- | --- |
| 行 | |
| 0 | 移到当前一行的开始[Home]  数字零，到行头 |
| $ | 移到当前一行的最后[End] |
| ^ | 命令将光标移动到当前行的第一个非空白字符上 |
| g\_ | 到本行最后一个不是blank字符的位置 |
| k | 光标上移一行Ctrl+p |
| j | 光标下移一行Ctrl+n |
| Enter | 光标下移一行 |
| n+ | 光标下移n行【按上档键 数字shift +】 |
| n- | 光标上移n行 |
| G | 移到文件的最后一行 |
| nG或者:n | 移到文件的第n行 |
| gg | 移动到文档的开始 |
| [[ | 文件开始位置——开始行 |
| ]] | 文件结束位置——末尾行 |
| H | 光标移至屏幕顶行 HEAD  光标定位在显示屏的第一行 |
| M | 移到屏幕的中间行开头 Middle  光标定位在显示屏的中间 |
| L | 移到屏幕的最后一行 LAST  光标定位在显示屏的最后一行 |
| ( | 光标移至句首 |
| ) | 光标移至句尾 |

|  |  |
| --- | --- |
| 段落 | |
| { | 移到段落的开头 |
| } | 移到下一个段落的开头 |
| % | 匹配括号移动，包括 (, {, [. （陈皓注：你需要把光标先移到括号上）  跳转到与之匹配的括号处 |
| \* 和 # | 匹配光标当前所在的单词，移动光标到下一个（或上一个）匹配单词（\*是下一个，#是上一个） |
| zf | 折叠（需加方向键） |
| zo | 展开（空格也可以展开）  可以zf进行折叠, 用zo打开折叠,也可以方向键向右打开折叠,zc 关闭折叠(只要在被折叠的块中的任一个语句就行)??? |

|  |  |
| --- | --- |
| 屏幕 | |
| Ctrl+u | 向文件首翻半屏up |
| Ctrl+d | 向文件尾翻半屏down |
| Ctrl+f 或[page down] | 向文件尾翻一屏 forward (fact整屏去两行) |
| Ctrl＋b 或[page up] | 向文件首翻一屏back (fact整屏去两行) |
| CTRL-] | 跳转到当前光标所在单词对应的主题 |
| CTRL-O | 回到前一个位置 |
| zz | 命令会把当前行置为屏幕正中央(z字取其象形意义模拟一张纸的折叠及变形位置重置) |
| zt | 命令会把当前行置于屏幕顶端(top) |
| zb | 命令会把当前行置于屏幕底端(bottom) |
| 50% | 光标定位在文件的中间 |
| ` | 跳转到最近光标定位的位置（只能记忆最近两个位置） 反引号 |

#### ****5.插入****

|  |  |
| --- | --- |
| i | 在光标前开始插入字符 insert |
| I | 在当前行首开始插入字符 |
| a | 在光标位置后开始加字 append |
| A | 在光标所在行的最后面开始加字 |
| o | 在光标下加一空白行并开始加字 open |
| O | 在光标上加一空白行并开始加字 |
| r | 替换当前字符 |
| R | 替换当前字符及其后的字符【当前及其后字符被覆盖】 |
| s | 默认删除光标所在字符，输入内容插入之  = xi |
| S | 默认删除当前行内容，输入内容作为当前行新内容= dd+o |

#### ****6.删除****

|  |  |
| --- | --- |
| 字符 | |
| nx | 删除由光标位置起始后的n个字符（含光标位置）  x =dl(删除当前光标下的字符) |
| nX | 删除由光标位置起始前的n个字符（含光标位置）  X =dh(删除当前光标左边的字符) |
| d0 | 删至行首 |
| d$ | 删至行尾 |
| dfa | 表示删除从当前光标到光标后面的第一个a字符之间的内容 |
| D | 代表d$(删除到行尾的内容) |
| C | 代表c$(修改到行尾的内容) |

|  |  |
| --- | --- |
| 单词 | |
| ndw | 删除光标处开始及其后的n-1个字 |
| ndb | 删除光标处开始及其前的n-1个字 |
| diw | 删除当前光标所在的word(不包括空白字符)，意为Delete Inner Word 两个符号之间的单词 |
| daw | 删除当前光标所在的word(包括空白字符)，意为Delete A Word |

|  |  |
| --- | --- |
| 行 | |
| ndd | 删除当前行及其后n-1行 |
| :n1,n2 d | 将 n1行到n2行之间的内容删除 |
| dG | 删除当前行至文件尾的内容 |
| dgg | 删除当前行至文件头的内容 |
| **d回车** | **删除2行【包括光标一行】** |

【删除就是剪切，它会覆盖之前复制的内容】

|  |  |
| --- | --- |
| 删除并进入输入模式 | |
| cw | 删除当前字，并进入输入模式  【很好用，快速更改一个单词】  相当于dw+i |
| ncw | 删除当前字 及其后的n-1个字，并进入输入模式\  修改指定数目的字 |
| cc | 删除当前行，并进入输入模式 |
| ncc | 删除当前行 及其后的n-1行，并进入输入模式 |
| guw | 光标下的单词变为小写 |
| gUw | 光标下的单词变为大写 |
| xp | 左右交换光标处两字符的位置 |
| ga | 显示光标下的字符在当前使用的encoding下的内码 |

#### ****7.复制和黏贴****

|  |  |
| --- | --- |
| 复制 | |
| nyl | 复制n个字符(也可nyh) |
| yw | 复制一个单词 |
| y0 | 表示拷贝从当前光标到光标所在行首的内容 |
| y$ | 复制从当前位置到行尾 |
| yfa | 表示拷贝从当前光标到光标后面的第一个a字符之间的内容 |
| yG | 复制从所在行到最后一行 |
| nyy | 将光标所在位置开始的n行数据复制暂存  复制一整行 |
| ctrl + v 方向 y | 复制选择的很多行：先使用V进入visual 模式，然后j向下移动到你想复制的行为止，然后y |
| 黏贴 | |
| p | 复制暂存数据在光标的下一行 |
| P | 复制暂存数据在光标的上一行 |
| :n1,n2 co n3 | 将n1行到n2行之间的内容拷贝到第n3+1行【n3行的下一行】 |
| :n1,n2 m n3 | 将n1行到n2行之间的内容移至到第n3行下 |
| J | 把下一行的数据连接到本行之后  多一空格 |
| ~ | 改变当前光标下字符的大小写 |

#### ****8.查找****

|  |  |
| --- | --- |
| /pattern | 从光标开始处向文件尾搜索pattern  【直接/，不需要先冒号】 |
| ?pattern | 从光标开始处向文件首搜索pattern |
| n | 在同一方向重复上一次搜索命令 |
| N | 在反方向上重复上一次搜索命令 |
| # | 向上完整匹配光标下的单词  相当于？word |
| \* | 向下完整匹配光标下的单词  相当于/word |
| % | 查找对应的( [ { 匹配 |
| nfx | 在当前行查找光标后第n个x（一般直接fx） |

#### ****9.替换****

|  |  |
| --- | --- |
| :s/p1/p2/g | 将当前行中所有p1均用p2替代  无g，则只替换第一个 |
| :s/p1/p2/c | 查找替换要求确认 |
| :n1,n2s/p1/p2/g | 将第n1至n2行中所有p1均用p2替代 |
|  |  |
| :%s/p1/p2/g | 全局，使用p2替换p1 |
| :%s/p1/p2/gc | 替换前询问 |
| :n,$s/vivian/sky/ | 替换第n行开始到最后一行中每一行的第一个vivian为sky，n为数字 |
| :.,$s/vivian/sky/g | 替换当前行开始到最后一行中每一行所有vivian为sky |
| :s/vivian\//sky\// | 替换当前行第一个vivian/为sky/，可以使用\作为转义符 |
| :1,$s/^/some string/ | 在文件的第一行至最后一行的行首前插入some string |
| :%s/$/some string/g | 在整个文件每一行的行尾添加some string |
| :%s/\s\+$// | 去掉所有的行尾空格，“\s”表示空白字符（空格和制表符），“\+”对前面的字符匹配一次或多次（越多越好），“$”匹配行尾（使用“\$”表示单纯的“$”字符） |
| :%s/\(\s\*\n\)\+/\r/ | 去掉所有的空白行，“\(”和“\)”对表达式进行分组，使其被视作一个不可分割的整体 |
| :%s!\s\*//.\*!! | 去掉所有的“//”注释 |
| :%s!\s\*/\\*\\_.\{-}\\*/\s\*!!g | 去掉所有的“/\* \*/”注释 |
| :%s= \*$== | 将所有行尾多余的空格删除(没看懂) |
| :g/^\s\*$/d | 将所有不包含字符(空格也不包含)的空行删除 |
| g/^$/d | 替换空行 |
| r | 替换当前字符 |
| R | 替换当前字符及其后的字符，直至按ESC键 |
| 命令说明： | |
| :[range]s/pattern/string/[c,e,g,i]   * range 指的是範圍，1,7 指從第一行至第七行，1,$ 指從第一行 至最後一行，也就是整篇文章，也可以 % 代表 (目前編輯的文章)。 * pattern 就是要被替換掉的字串，可以用 regexp 來表示。 * string: 將 pattern 由 string 所取代。 * c: confirm，每次替換前會詢問。 * e: 不顯示 error。 * g: globe，不詢問，整行替換。 * i: ignore 不分大小寫。 * g 大概都是要加的，否則只會替換每一行的第一個符合字串。 * 可以合起來用   其中s为substitute，%表示所有行，g表示globa | |

#### ****10.编辑多个文件****

|  |  |
| --- | --- |
| :r filename | 将指定文件的内容读入光标所在行下 |
| :args | 显示编辑名单中的各个文件名列表 |
| :n | 切换到下一个文件 |
| :N | 切换到上一个文件 |
| :rew | 回到首个文件 |
| :e# | 读入编辑名单内的前一个文件 |
| :e file | 读入另一个文件进vi(此文件可不在编辑名单内）,若原文件经修改还没有存档，则应先以: w 存档。 |
| :e! file | 强迫读入另一个文件进入vi，原文件不作存档动作。 |
| :e | 放弃当前一切修改，重新载入文件 |
| :bn 和 :bp | 你可以同时打开很多文件，使用这两个命令来切换下一个或上一个文件 |
| :files 或 :buffers 或 :ls | 会列出目前 buffer 中的所有文件 |

#### ****11.其他命令****

|  |  |
| --- | --- |
| . | 重复前一指令 |
| u | 取消前一指令undo  :u也行，一般不用，操作太多 |
| Ctrl + r | 恢复【只对u有效】redo |
| Ctrl + l | 刷新屏幕显示 |
| Ctrl+v  然后 ctrl+A是^A  Ctrl+I是\t | 输入特殊字符 |
| Ctrl+v  然后用j、k、l、h或方向键上下选中多列，之后 I I a A r x y d等，最后按esc，生效 | Vim列操作 |
| V | 进入visual 模式【ESC】退出  多行选中模式 |
| Ctrl+z挂起退出，fg恢复 |  |



【set 可以简写为se】

附录：set的所有选项

|  |  |
| --- | --- |
| :set all | 列出所有选项设置情况 |
| 常用 | |
| :se number  简写 :se nonum | 显示文件的行号  简写:se nu se nonu |
| :se list | 显示制表位(Ctrl+I)和行尾标志($) se nolist |
| :se wrap | 将超出屏幕行分多行显示 se nowrap  打开/关闭 换行 |
| :se paste | vim贴代码的时候，格式乱掉  先之后，黏贴，就不会乱掉 |
| :set ignorecase | 设置在搜索中忽略大小写 |
| :set hlsearch | 高亮显示搜索结果 |
| :nohls | 可以取消高亮 |
| 其他 | |
| term | 设置终端类型 |
| report | 显示由面向行的命令修改过的数目 |
| terse | 显示简短的警告信息 |
| warn | 在转到别的文件时若没保存当前文件则显示NO write信息 |
| nomagic | 允许在搜索模式中，使用前面不带“\”的特殊字符 |
| nowrapscan | 禁止vi在搜索到达文件两端时，又从另一端开始 |
| mesg | 允许vi显示其他用户用write写到自己终端上的信息 |
| :set autowrite | 设置自动存盘 |
| :set backup | 设置备份 |
| :set backupext=.bak | 设置备份文件名后辍，默认原文件名~ |
| :syntax enable | 打开彩色 |
| :set autoindent | 让Vim在开始一个新行时对该行施以上一行的缩进方式 |
| :set showcmd | 在Vim窗口的右下角显示一个完整的命令已经完成的部分 |
| :set key= | 去掉文件加密 |
| :set ai | 设置每行起始位置（以光标当前位置为起始） |
| :set noai | 取消行起始位置设定 |
| :set incsearch | 键入目标字符串的过程中Vim就同时开始了搜索工作 |
| :set ts=4  :set expandtab | 一个Tab自动转换成4个空格 |
| :syntax on/off | 颜色，简介:syn on/off |

#### ****12分屏操作****

|  |  |  |
| --- | --- | --- |
| 分屏启动Vim | | |
|  | | |
| vim -On file1 file2 ... | | 使用大写的O参数来垂直分屏 |
| vim -on file1 file2 ... | | 使用小写的o参数来水平分屏。 |
| **注释:** n是数字，表示分成几个屏【只会出现n屏，后面跟多了文件不会显示】  使用VIM的帮助 :help split | | |
| 分屏 | | |
| Ctrl+W s | 上下分割当前打开的文件 | |
| Ctrl+W v | 左右分割当前打开的文件 | |
| :sp filename | 上下分割，并打开一个新的文件 | |
| :vsp filename | 左右分割，并打开一个新的文件 | |
| 关闭分屏 | | |
| Ctrl+W c | 关闭当前窗口【关闭不了最后一个】 | |
| Ctrl+W q | 关闭当前窗口，如果只剩最后一个了，则退出Vim | |
| :qa | 退出vim，并关闭所有屏 | |
| :wqa | 保存所有分屏并关闭 | |
| 移动光标 | | |
| Ctrl+W l | 把光标移到**右边**的屏 | |
| Ctrl+W h | 把光标移到**左边**的屏中 | |
| Ctrl+W k | 把光标移到**上边**的屏中 | |
| Ctrl+W j | 把光标移到**下边**的屏中 | |
| Ctrl+W w | 把光标移到**下一个**的屏中 | |
| 要在各个屏间切换，只需要先按一下Ctrl+W  <C-w><dir> : dir就是方向，可以是 hjkl 或是 ←↓↑→ 中的一个，其用来切换分屏 | | |
| 移动分屏 | | |
| Ctrl+W L | 向右移动 | |
| Ctrl+W H | 向左移动 | |
| Ctrl+W K | 向上移动 | |
| Ctrl+W J | 向下移动 | |
| 这个功能还是使用了Vim的光标键，只不过都是大写。当然了，如果你的分屏很乱很复杂的话，这个功能可能会出现一些非常奇怪的症状。 | | |
| 屏幕尺寸 | | |
| 下面是改变尺寸的一些操作，主要是高度，对于宽度你可以使用Ctrl+W <或是>，但这可能需要最新的版本才支持  当同时打开几个文件时，按 \_ 使当前窗口最大化 | | |
| Ctrl+W = | 让所有的屏都有一样的高度 | |
| Ctrl+W +【shift +=键】 | 增加高度  <C-w>+ (或 <C-w>-)  增加尺寸  扩大窗口 | |
| Ctrl+W -【shift –键】 | 减少高度  缩小窗口 | |
| <C-w>\_ (或 <C-w>|) | 最大化尺寸 (<C-w>| 垂直分屏) | |

关闭分割窗口可以用:close 其实用:q也行

#### ****13 vim的帮助指令 &警告信息****

Vim 拥有一个细致全面的在线帮助系统,进入帮助

|  |
| --- |
| 启动帮助 |
| <HELP> 键 (如果键盘上有的话) |
| <F1>键(如果键盘上有的话) |
| :help<回车> |
| :help command<回车>  例如:help w <回车>  :help insert-index <回车> |
| 退出帮助 |
| :q <回车> |

|  |  |
| --- | --- |
| 警告信息 | |
| [O]pen Read-Only | 打开此文件成为只读文件 |
| [E]dit anyway | 正常方式打开，不会载入暂存文件的内容 |
| [R]ecover | 加载暂存文件 |
| [D]elete it | 打开文件前删除暂存文件 |
| [Q]uit | 退出，不打开 |
| [A]bort | 忽略这个编辑行为 |

##### VIM操作模式转换图

补充：从命令模式到底行模式，可以直接 : ?两个搜索符号，不需要先按冒号

#### ****14 七个习惯****

* + 快速移动
  + 不要两次键入同样的东西
  + 错误修复
  + 经常需要编辑不止一个文件
  + 协同作业
  + 文本是结构化的
  + 养成习惯

1. 快速移动

在文本中随意漫游是非常常见的操作。所以高效编辑的第一要义是学习如何能够在文本中快速移动, 准确定位。有三个步骤可以使你学到你需要的技巧:

当你编辑文件的时侯, 留意一下你经常要重复进行的操作是什么；

练习使用这些命令, 直到你的手指可以不假思索地运用自如；

浏览一下参考手册你就会发现关于tag的主题。文档会告诉你如何使用这一功能跳转到函数的定义处。

2. 不要两次键入同样的东西

借助":s"替换命令或者"."重复命令等快捷操作代替重复操作。

3. 错误修复

缩写功能来纠正。一些例子是::abbr Lnuix Linux。

语法高亮机制。

4. 经常需要编辑不止一个文件

创建tag文件。

将光标定位在你要查看其原型的函数名上, 然后按下"[I"命令, Vim将会显示include文件中匹配这个函数名的一个清单。

预览标签机制会打开一个特殊的预览窗口, 并且使光标仍然停留在你当前所在的位置，在预览窗口中的文本列出了当前光标所在处的函数的声明(有些可能不是声明) 将当前光标移动到另一个函数名上, 停留几秒钟, 预览窗口中的内容就会变成是关于新函数名的声明。

5. 协同作业

Unix的哲学是使用独立的小程序, 每个小程序做一项专门的任务,并且把它作好, 将它们的工作整合到一起来完成一个复杂的任务。

Vim的做法是将一些分散的小程序整合起来, 产生强大的处理能力,在这方面Vim还有待在将来进一步提高。

在编辑器领域, emacs是这方面的一个典范(有人甚至说它是一个能编辑文本的操作系统)

6.文本是结构化的

加速你的编辑-编译-修改的周期。

":make" 命令编译程序项目, 捕获编译的错误/警告并允许你直接跳转到引起这一错误/警告的程序行上去；

`errorformat'选项告诉Vim你的编译器将生成何种格式的错误信息, 以便于它能识别；

写一些自定义的宏, 如要跳转到manual 帮助文档, 这对于查看交叉索引是一种简捷有效的办法；

使用上面的三项原则可以对付任何形式的结构化文本。

7. 养成习惯

绝大多数人只需要学习其中的10-20%的命令就足以应付它们的工作了。

建立适合自己的命令集, 不时地回顾以往所做的事, 看看是不是可以自动完成一些重复的工作。

不断地重复练习你的解决方案直到你的手指可以条件反射地自动完成, 从而达到你所期望的境界。

不要一次尝试太多的东西, 一次做一件事并多做几次会好得多。

对于不经常的操作, 最好记下你的处理步骤以备将来不时之需。

不管怎样, 只要目标明确。你就能找到让你的编辑变得更加高效的办法。

更多的是技能而非知识

更多的是双手而非头脑