

# Vim7使用手册

简体中文版  
(手册版本号: 0.5)  
翻译: BillPeng

---

# Vim中文版：目录表

---

首页

翻译

1. 前言
2. Vim介绍
3. Vim安装
4. Vim起步
5. Vim模式
6. 文字输入技巧
7. 搜寻定位方法
8. Vim帮助
9. 基础编辑命令
10. 更多的编辑命令
11. 文件内容的多重编辑
12. 个人信息管理
13. 脚本编辑
14. 脚本插件系统
15. 程序编辑
16. 关于编程的更多信息
17. 进一步学习
18. 信息反馈
19. 慈善基金软件
20. 版本说明

---

前页 后页

手册出自于：[http://www.swaroopch.com/mediawiki/index.php?title=Vim\\_en:Table\\_of\\_Contents&oldid=1053](http://www.swaroopch.com/mediawiki/index.php?title=Vim_en:Table_of_Contents&oldid=1053)

首要作者：Swaroop

---

## 关于Vim

Vim是一个用来编辑文件的计算机软件，且提供了大量帮助你编辑的功能。

## 为什么用Vim?

在现实之中，其实写出好的作品很少第一次就能够成功。最可能的就是你们不断的编辑直到写的更好为止。正如Louis Brandeis曾经说的："没有伟大的写作，只有伟大的修改。"假如有一个功能齐全的编辑器来帮助我们的话，快速大量的修改将会变的非常容易，而这正是Vim闪光的地方，并且与之相比较，远远好于大多数一般文本和富文本编辑器。

## 我为什么要写这本手册呢？

自从我在大学学习unix课期间使用老式vi编辑器以来，我就一直在用Vim了。它是少有的几个我一天能用近10个小时的软件。我知道它有那么多的功能，我却不了解，但是对我以后却很有用，因此我就一点一点的慢慢揭开它的面纱。为了使我的理解更加具体、牢固，也为了帮助他人学习Vim，我开始写学习笔记，这本笔记的集合就是所谓的Vim手册了。

在我写这些笔记的时候，我总是坚持我的几条原则，如下：

1. 只是一些简单的笔记，这一点的重要性一定要加强再加强。
2. 着重于举例、使用方法。
3. 为阅读者使用一条龙服务 - 从最初运行直到学习高级操作指南。
4. 让用户理解怎么按照Vim的方法来编辑-从模式编辑到缓冲区编辑直到随心所欲。但绝大多数的用户只学了基本的Vi命令，除此以外就不在学其他任何东西。如果按照上述方法的话，或许这是个转折点。他们将变成铁杆的Vim用户。Vim用户，就是渴求充分使用Vim功能完成工作，这也就是我写这本手册的初衷。
5. 这本手册里包含了大量所需代码内容给用户做参考。比如怎么把Vim环境做成集成开发环境（IDE），虽然有各种各样的方法，但为了使读者不至于辛辛苦苦想出哪个脚本插件来做这件事，手册里已经为你们准备了基本的运行代码。
6. 我就给还算多的笔记给读者理解使用，而不是全部。（帕拉图的原则）
7. 与之相关的是，这本手册不应该改写手册中的参考指南，而应该把手册中的有关指南部分标出来。这样，就没有重复。

要是用户能够使用非常棒的内建参考指南的话--这点很重要，那么这本手则的优势也就能够长盛不衰。

总结一下，我写的要诀就是概念、举例、简洁。

## 手册维护情况

目前，这本手册正在持续维护之中。按照现在的维护情况，还不能算是在为“1.0”版本做准备。

在手则维护的时候，非常欢迎建设性的意见，提议。如有好的建议和想法请通过官方网站网页左边侧栏 'Discussion' 按钮联系我们，或者Email给我。

## 官方网址

手则的官方网址是 <http://www.swaroopch.com/notes/Vim>。在网站上，你能够在线阅读正本手册和最近的手册开发版本，还可以发送反馈信息给我。

1. 本手册在the Creative Commons Attribution-Share Alike 3.0 Unported license下许可使用。

用户拥有的权利：

用户享受自由拷贝、分发、传播作品的权利。

用户享受自由掺合改写作品代码的权利。

用户拥有的义务：

归属性。用户必须以作品作者或者许可证颁发者指定的方式归属作品的拥有者。（但并不是在任何方面的建议、使用都能被认可。）

平等性。假如用户修改、改变、或以作品为基础，只能在与作品相同或相似的许可证下分发新作品。

任何作品重复使用或分发，用户必须使其他用户明白该手册的许可条例。

假如用户得到版权所有者的授权，上述任何限制条款均对用户无效。

许可证的任何条款对作品作者无效。

2. 该手册归属性必须通过链接到<http://www.swaroopch.com/notes/Vim>网页来显示，而且清楚的指明从本网页能够得到原始作品。

---

3. 该手册中的所有代码和脚本都在the 3-clause BSD License许可证下使用，另有说明除外。
4. 该手册中的样文在GNU自由文档许可证下撷取于<http://en.wikipedia.org>和<http://en.wikiquote.org>网站。
5. 对该手册的原始版本有贡献的自愿者必须受限于相同的许可证，而且版权归主要作者所有。

## 一些思考

手册没有完工--他们还在修改，手册里面也有你们自己的代码，这让人难以接受，尤其是手册的第七次修改还不是很完美。

-- Michael Crichton

完美的作品，不是在修修补补中诞生，而是出自作品整体完整性。

-- Antoine de Saint-Exupery

---

前页 后页

### 相关的外部网站

[1] <http://www.vim.org>

[2] <http://www.swaroopch.com/contact/>

[3] <http://creativecommons.org/licenses/by-sa/3.0/>

[4] <http://www.opensource.org/licenses/bsd-license.php>

[5] <http://en.wikipedia.org/wiki/>

维基百科全书：Text\_of\_the\_GNU\_Free\_Documentation\_License

内容出自于：[http://www.swaroopch.com/mediawiki/index.php?title=Vim\\_en:Preface&oldid=1015](http://www.swaroopch.com/mediawiki/index.php?title=Vim_en:Preface&oldid=1015)

首席作者：Swaroop

## Vim中文版：Vim介绍

---

Vim是什么？

---

Vim编辑器是一款计算机软件，主要是帮助用户编辑文档，对任何文档编辑者来说是个福音--比如写张购物单、一本书或者软件代码什么的。Vim编辑器的特色在类似的几款软件之中算是即操作简单且功能强大。

简单意味着Vim编辑器很容易上手，意味着用户利用很小的界面环境就能专心致志的做重要的工作--写作，也意味着Vim的核心理念就是让用户很容易的学习高级功能。

功能强大意味着Vim编辑器编辑效率高、质量好并且操作更容易，意味着编辑看似简单的文档成为可能，意味着Vim编辑器不是复杂难学，意味着Vim编辑器一直以“最少的努力，最大的成果”为用户使用目标。

## Vim编辑器能做什么工作？

或许我能听到有的用户说：“不就是个文档编辑器嘛，难道还有什么大不了的吗？”

说的对，vim确实有很多丰富的功能。

让我们随便看些实例，与你们正在使用的编辑器来个对比。对于所举的例子而言，它们就是明了的答案--"在我使用的编辑器中，我怎么才能实现此项功能呢?"

说明

示例中命令的详细细节不要太过于关注，它的目的是近可能的对你有所启发。而不是现在就解释命令的用法，这是手册的其他章节要陈述的。

要做的事	命令
怎样才能使光标向下移动7行？	按7j两键
怎样删除一个字？	按dw两键
是的，就一个"字"？	
怎样全文搜索光标处的这个字？	按*键
怎样查找替换50-100行中间的字？	运行:50,100s/old/new/g
如果想同时查阅同一个文档的两个不同部分，应该怎样做呢？	运行:sp 水平分割浏览
如果想打开一个文件，文件名就在当前的文件夹里并且光标那里处的文字就是文件名，这怎么做呢？	按gf两键(意思就是'g'o去文件'file')

假如想选择一种喜好的颜色显

示方案，这应该怎么做呢？

如果想映射键盘快捷方式`ctrl-s`来储存  
文件内容，该怎么做呢？

如果想存储当前所有打开的文件

的设置并且任何文件的设置都已经改变，为了以后  
能继续编辑应该怎么做呢？

如何想不同的颜色相对应不同的  
代码，应该怎么做呢？

假如想把文档的其他部分

隐藏收起来，一次只专心编辑其中  
一部分，该怎么做呢？

假如想打开不同的标签，编

辑不同的文件，该怎么做呢？

假如文档中有一些频繁使用的字句，想在下次  
需要输入的时候，快速补上，该怎么做呢？

假如有一些文档中的数据，只想要每行的头十  
个字符，其他的不要，怎么才能得到要用的文  
本呢？

运行：`colorscheme desert`，来选

择'`desert`'颜色显示方案，我最喜欢的。运行：`nmap <c - s> :w<CR>`，注意  
<CR>的意思就是'`c`'arriage '`r`'eturn  
即回车键。

运行：`mksession ~/.latest_session.vim`，下次打开的时候用`Vim -S ~/.latest_session`

运行：`syntax on`，假如不能正确识别所用的语言话，就设置`filetype=Wikipedia`，举个例子。

运行：`set foldmethod=indent`，假设你的文档设为`indent`模式，除此外还有其他办法。

运行：`tabedit <文件>`，打开多个标签，编辑多个文件，非常类似于浏览器的标签功能，用`ctrl-pgup` 和`ctrl-pgdn`来切换不同的标签。

按`ctrl-n`看当前“完成”清单里的字句，这些都是文档里用过的。当在正文中输入`m a s <space>`的同时有选择地在命令行键入：`:ab mas Maslow's hierarchy of needs to expand the abbreviation automatically.`

按`ctrl-v`选择你所要文本，然后按`y`键拷贝这些数据就行了。

假如从别人那里收到一个文档，全是  
大写，看起来有点烦怎么才能  
把内容都改成小写的呢？

在Vim命令行中，运行如下代码：

```
:for i in range(0,  
    line('$'))  
: call setline(i,  
    tolower(getline(i)))  
: endfor
```

代码执行的细节问题，在下面的章节里将会一一展开。 这里还有一个更简练的命令，运行：`%s#\(.\\)\#\1\\1#g`。但是话又说回来，似乎更容易想到用上面的代码来改大小写。其实，最简单的办法就是选择整篇文档，按u键就可以了。但是这样还是不太酷，不能炫耀Vim一步步执行代码的成就感。

好了，阅读到这里，Vim的功能令你信服了吗？

在上面的示例中，能够看到Vim编辑器功能强大的编辑能力。任何一款其他的编辑器哪怕再怎么折腾都很难有Vim这样的功能。

然而，令人吃惊的是，所有的功能都非常的通俗易懂。

值得注意的是，在编写这些示例的时候，我们一次都没有用过鼠标！这是个好现象，有兴趣，算一下手一天内在鼠标、键盘之间移动了多少次，你就会发觉其实不用鼠标感觉不错哦。

不要被这里的命令搞的晕头转向，Vim最好的地方就是不需要熟知它自带的的所有功能。用户只需要知道几个最基本的概念，有了这么几个基本的概念，其他所有的功能在需要的时候轻轻松松的就学会了。



内容来之于: [http://www.swaroopch.com/mediawiki/index.php?title=Vim\\_en:Introduction&oldid=1102](http://www.swaroopch.com/mediawiki/index.php?title=Vim_en:Introduction&oldid=1102)

首席作者: Swaroop

## Vim中文版:Vim安装

---

让我们看看怎么把Vim编辑器安装在计算机上。

### Windows操作系统

如果你使用windows, 然后按照下面的步伐将会帮助你获得Vim的最新版本, 安装在你的计算机上:

1. 访问网站<http://www.vim.org/download.php#pc>
2. 下载 "Self-installing executable" (gvim72.exe [1] 截至到写书的时候)
3. 双击这个文件安装Vim, 就如安装其他的应用程序一样。.

### Mac OS X苹果操作系统

如果你用Mac OS X, 并且已经装了Vim编辑器的终端版本。 打开菜单命令程序 功能终端。 在终端中, 键入Vim命令回车, 你就能看到Vim经典的欢迎屏幕了。

如果你想使用Vim的图形桌面版本, 请打开Cocoa-based MacVim project [2], 下载最新的版本。

接着双击安装包 (比如: MacVim-7\_2-stable-1\_2.tbz), 包就会被打开建立一个 vim72 的目录。进入目录, 拷贝gvim和vim应用程序到你自己的目录下面就行了。

### 疑惑

Mac OS X 用户能证实一下这样的操作还能用吗?

### Linux/BSD操作系统

如果你正在用某个Linux或者哪个\*BSD的发行版, 你可以直接使用已经安装好的Vim版本。

---

打开终端，比如：`konsole`或`gnome-terminal`，运行`vim`命令就能看到Vim欢迎屏幕了。

假如你得到类似的信息-`vim: command not found`，那就是Vim没有安装，你就的用系统安装工具安装Vim了。

比如Ubuntu/Debian发行版的`aptitude`工具，小红帽Fedora的`yum`工具，FreeBSD系统的`pkg_add`和`port`工具等等。

具体情况请参考相关的系统文档和论坛上的帖子。

需要安装图形版的话，用`vim-gnome` package包或选择其他的Vim包。

## 总结

安装好了以后，你就能在终端中运行Vim了，要么打开系统菜单调用Vim应用程序。

既然已经把vim安装在系统上面了，接下来我们将继续讲解怎么使用Vim。

---

[前页](#) [后页](#)

[外部链接](#)

[1] <ftp://ftp.vim.org/pub/vim/pc/gvim72.exe>

[2] <http://code.google.com/p/macvim/>

内容来自于：[http://www.swaroopch.com/mediawiki/index.php?title=Vim\\_](http://www.swaroopch.com/mediawiki/index.php?title=Vim_)

[en:Installation&oldid=1103](#)

首席作者：Swaroop

## Vim中文版:Vim起步

---

### 开始学习Vim

首先，当然是学会启动Vim。

### Vim图形版

### Windows系统

---

点开始菜单      程序菜单

Vim 7栏      点gVim图标。

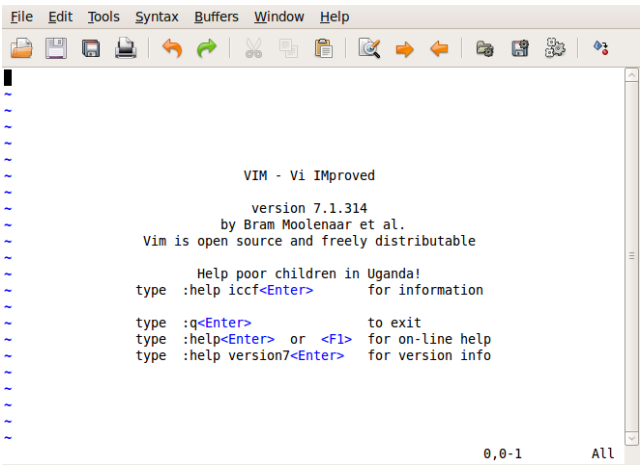
Mac OS X系统

点应用程序菜单      Vim图标。

Linux/BSD系统

点应用程序菜单

附件      GVim文本编辑器，  
或者按Alt+F2，输入gvim，回车。



## Vim终端版

Windows系统

点开始      运行,输入vim，回车。

Mac OS X系统

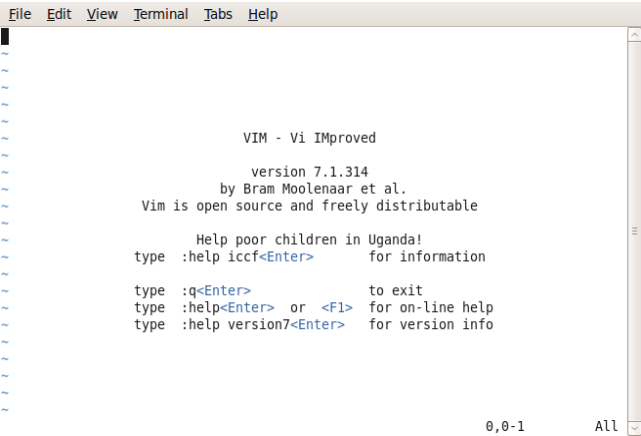
点应用程序菜单      功能栏

终端，输入vim，回车。

Linux/BSD系统

点应用程序菜单

附件栏      终端,或者按Alt+F2,输入konsole/gnome-terminal，回车。打开终端后，输入vim，回车。



从先起,我们所用的 ‘ open vim ’ -打开Vim，就是用上述的两种方法。

## 注意：

当启动Vim的时候，也许你会注意到不能马上输入正文，请不要惊慌，一会就知道怎么做了。

## Vim选择图形版，还是终端版？

图形版的Vim在其应用窗口的顶部有菜单，只要操作鼠标就可以打开不同的选项。但用鼠标不是唯一的选择，只用键盘也能使用全部的功能。

为什么操作键盘如此重要呢？因为一旦有人想提高输入效率，就用键盘的话，打字可以更快，出错概率更小，正好与用鼠标相反。这是因为手在鼠标和键盘之间不停的切换很慢。并且人们的思维也需要一个切换过程，在手，键盘，鼠标切换的时候。如果我们用键盘快得可以像只兔子，就可以节省很多不必要的切换时间。

当然，这只是主观上的想法。有些人喜欢鼠标，而有些人喜欢键盘。而我鼓励你用键盘，才能真正的体会到Vim的威力。

## (Modes)模式介绍

想像今天是星期六的夜晚，你正在对电视上播出的电视剧抱怨，反而想看一看经典的老电影。因此，你切换TV到电影模式，由此DVD播放的画面替代了有线电视的信号。值得注意了，电视剧仍然在播放，但是你切换了看DVD影视还是看电视生活频道的场景。

想像今天是星期六的夜晚，你正在对电视上播出的电视剧抱怨，反而想看一看经典的老电影。因此，你切换TV到电影模式，由此DVD播放的画面替代了有线电视的信号。值得注意了，电视剧仍然在播放，但是你切换了看DVD影视还是看电视生活频道的场景。

差不多，Vim也有模式。例如，Vim可以用一个模式写作，一个模式运行命令。不过这些模式都与文档编辑有关，而你却切换了是继续输入文字还是运行命令的情景。

这个概念不容易理解，不是吗？

一般来说，模式的概念是让新手感到Vim含混不清的最显著原因。我把学Vim比作骑自行车 - 你总会摔几次，但一旦掌握好龙头，就会感叹原来是这样子啊。

为什么Vim要设计这样的模式？还是为了一切都尽可能的简单化，即使用户刚开始的时候感到很陌生。

我刚才说的是什么意思呢？我们举个例子吧- Vim设计的关键一点就是尽可能的从键盘读取信号，从来没有考虑过使用鼠标(要是你想用鼠标的话仍可以使用，但这仅仅是个可选的输入设备而已)。

在这样的案例中，你会怎么来辨别那些是需要书写的内容，那些是你要运行的命令呢？Vim'的解决办法是设计了一个"normal"(普通模式)，在这种模式中，你可以执行命令。另一个"insert"(插入)模式，在这种模式下，你可以正常的书写文档。任何时候都能随便切换两种模式。

比如，按i键切换到插入模式，按<Esc> 键切换到正常模式。正统的编辑器怎么在文本和命令之间达到这种效果呢？

通过图形菜单和快捷键。不过，这种方式无任何收缩性可言。其一，假如你有成百上千的命令，为每个命令建立菜单很疯狂，容易混淆视听。其二，定制命令的使用方式更加困难重重。

我们举一个具体的例子。假如你想改变所有出现在文档中的"from"到"to"之间的文字。在正统的编辑器中，你可以运行菜单命令。

就象 Edit -> Replace (或者快捷键如Ctrl-R)，然后输入从"from"到"to"间的文字 点"Replace"，在"Replace All"项上打勾就行了。

而在Vim的"normal"模式中，你只需简单的运行:s/from/to/g，:s是"substitute(替换)命令。看，多简单。

假如你想替换文档中的头十行，而且每次替换都用yes/no来确认，该怎么做呢？对于用正统的编辑器，达到这种效果很容易，通过取消'Replace All'选项就可以了。但注意到没有，你得先找到这个选项，然后用鼠标去点击它(或者用键盘敲一长串的字符)。然而怎么才能替换前10行呢？在Vim中，只要简单的运行:0,10s/from/to/gc就ok了。我们用的c选项就是'c'confirmation(确认)每次的替换。

---

虽然分开了写作模式(insert)和命令模式(normal), Vim却使我们在两种模式中换来换去非常容易。看吧, 刚开始学用Vim的时候确实有点奇怪而又有点陌生, 一旦你明白了它的作用, 意义就开始展现了。最好的, 就是这些核心理念将帮助你理解Vim命令的所有功能。从现在起, 你差不多理解了normal模式和insert模式区别, 你完全可以自己查找在normal模式使用的各种命令, 立即运行。与Vim相比, 正统的编辑器学习新得命令常常必须阅读大量的文档, 搜索大量的菜单, 不断的测试。要不就直接向高手请教。

个人而言, 我觉的模式的名字对新手不是很直观。我情愿把“insert”模式改成“writing”模式, “normal”模式改成“rewriting”模式。但我们得继续使用这些Vim专用术语以免引起混淆。

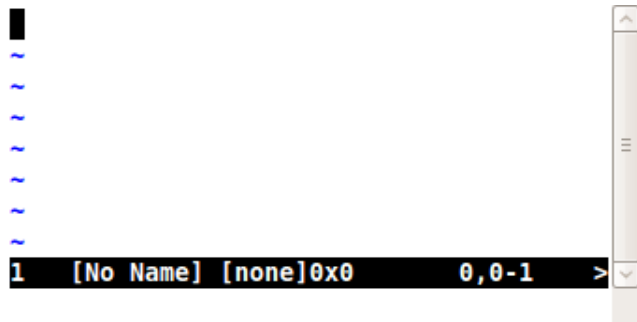
## 注意：

命令模式种的所有命令输入完后必须按回车键, 说明我们已经完了整个命令。象这样, 运行:help vim-modes-intro, 就表示你应当输入:help vim-modes-intro 后按回车键。

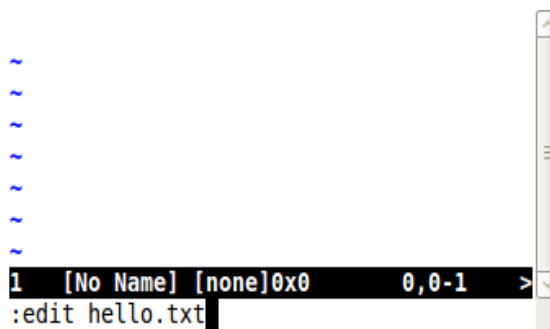
## 写一个文档

我们已经明白了打开和关闭文件的命令, 现在我们来作一些工作, 就是, 写作：

1. 打开Vim。

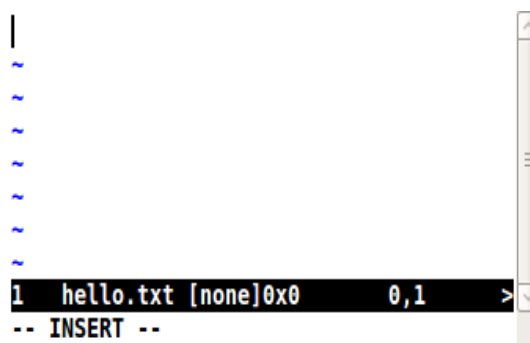


2. 输入:edit hello.txt ,回车。



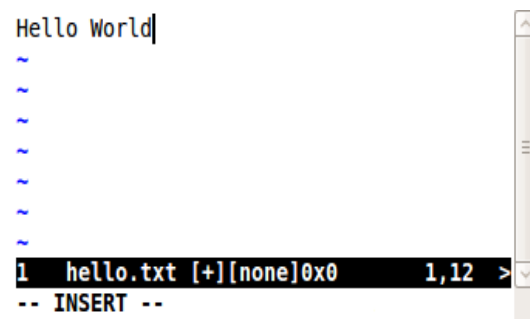
A screenshot of the Vim editor interface. The command line at the bottom shows the command `:edit hello.txt` entered. The status bar at the bottom right displays `1 [No Name] [none]0x0 0,0-1 >`. The editor buffer is empty, with blue wavy lines indicating the end of the file.

3. 按i键。



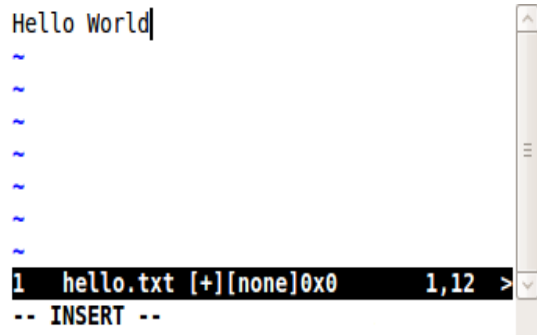
A screenshot of the Vim editor interface. The command line at the bottom shows `-- INSERT --`. The status bar at the bottom right displays `1 hello.txt [none]0x0 0,1 >`. The editor buffer is empty, with blue wavy lines indicating the end of the file.

4. 输入文本Hello World。



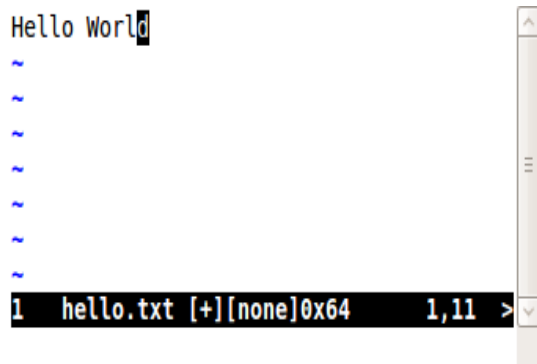
A screenshot of the Vim editor interface. The text `Hello World` has been entered in the editor buffer. The command line at the bottom shows `-- INSERT --`. The status bar at the bottom right displays `1 hello.txt [+] [none]0x0 1,12 >`. The editor buffer is empty, with blue wavy lines indicating the end of the file.

## 5. 按&lt;Esc&gt;键

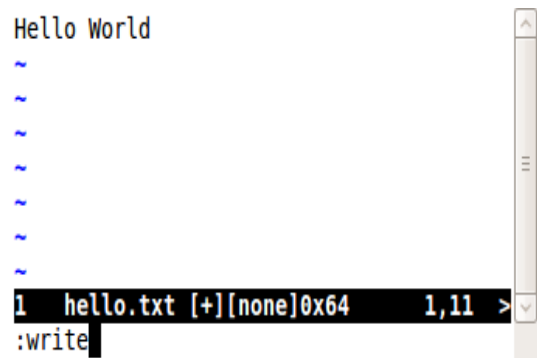


```
Hello World|
~
~
~
~
~
~
1 hello.txt [+] [none] 0x0 1,12 >
-- INSERT --
```

## 6. 输入:write, 回车。



```
Hello World|
~
~
~
~
~
~
1 hello.txt [+] [none] 0x64 1,11 >
-- INSERT --
```



```
Hello World
~
~
~
~
~
~
1 hello.txt [+] [none] 0x64 1,11 >
:write
```



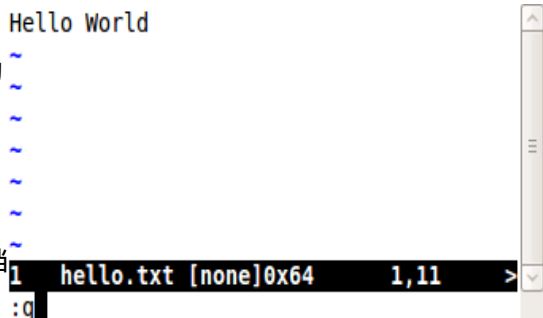
7. 关闭Vim, 输入:q, 回车。

恭喜哦!你刚才编辑了第一个文档,用的就是Vim。:-)

这看起来似乎有很多步?

是的,开始是这样。 那是因为首次使用

Vim就是建立文档,输入内容,关闭文档的过程。



你的想着,与真正编辑内容的时间相比,这只是很少的一点时间了。让我们看看命令都做了点什么事。

:edit hello.txt或者简化为:e hello.txt。

这个命令打开一个文件编辑。如果这个文件的名字没有指定,那么会在第一次存储的时候加上。

按i键

i键将切换到insert mode文本编辑模式

输入文本:Hello World

这里就是你想编辑文档内容的地方。

Press <Esc>

This escapes Vim back to normal mode

:write或者简化为:w

这个命令告诉Vim存储文件(目前存储在电脑的内存当中)到硬盘上去,意味着不管目前写了什么内容,都被永远的保存了。

:quit 或者简化为:q,退出Vim的编辑"window(窗口)".如果只有一个"window"在用,命令将会关闭vim(窗口的概念在以后的章节中讨论)。试着用不同的文档名称,不同的内容重复几次,才能习惯这些Vim的最基本的命令集。

注意一下，当你正在使用(insert mode)编辑模式的时候，Vim 会在窗口的左下角显示--INSERT（插入）--字样。而你切换到（normal mode)命令模式后，不会显示什么东西。那是因为Vim的缺省模式就（normal mode)是命令模式。

花点时间再琢磨琢磨这些内容，万事开头难嘛，也是为以后的内容下坚实的基础。：)

不要急，你需要的帮助信息就在你面前--实际上，运行:help命令就行了.比如，运行:help :edit你就会看到在窗口里面显示的帮助文档。就这样，试试吧。

## 总结

我们现在已经讨论了Vim最基本的概念和用法。建议看看:help notation说明文档和:help codes按键说明。

一定要很好的理解这些概念。一旦你开始了 "以Vim的方式思考问题",其他的Vim功能就很好理解了。

---

前页 后页

内容来自于: [http://www.swaroopch.com/mediawiki/index.php?title=Vim\\_en:First\\_Steps&oldid=1021](http://www.swaroopch.com/mediawiki/index.php?title=Vim_en:First_Steps&oldid=1021)

首席作者: Swaroop

---

## Vim中文版: 模式

---

### 模式介绍

在上一章, 我们已经接触到了模式 (Modes)。现在, 让我们进一步揭开模式的面纱- 关于可用模式的类型和每个模式类型的用途。

### 模式的类型

Vim中有三种基本的模式类型-normal (命令模式), insert (编辑模式) 和visual (可视模式)。

Normal mode (命令模式) 是指命令运行环境, 每次开启Vim的时候, 作为缺省模式应用。

Insert mode (编辑模式) 是指文本输入环境, 用插入等功能输入文本。

Visual mode (可视模式) 是指对一大块可选择的文本中的任何部分进行命令/操作的直观环境。

### Normal mode (命令模式)

缺省地, 你是在命令模式。让我们看看在此模式中能干点什么吧。输入:echo "hello world", 回车。你会看见经典的“hello world”语句回显给你。你刚才做得就是运行了一个Vim命令:echo, 并且你提供了内容给它, 马上就回显给你; 输入/hello, 回车,Vim就会搜索这个词组并且跳到第一次出现的地方。这只是这类命令中的两个简单的例子, 在normal mode (命令模式) 中很适用。在以后的章节中, 将会看到很多这样的命令。

### 如何使用帮助文档

它几乎与normal mode (命令模式) 一样重要, 一定要学会用:help命令。在帮助文档里, 你将学会更多的比较实用的命令。记住, 你完全没有必要通晓所有的实用命令。Vim的这种方式好的多, 帮助文档在你需要的时候, 会容易找到的你需要的命令详细内容。比如, 你运行:help usr\_toc命令, 它将把我们带到参考目录; 运行:help index 将会搜索你感兴趣的话题, 举个例子, 运行/insert mode 将会看到与insert mode相关的帮助信息。

如果你开始没有记住这两个主题, 只需要按F1或者简单运行:help。

## 插入模式

当Vim启动进入命令模式,我们直接用i键就可以切换到文本输入模式了。 也有其他的方法从命令模式切换到文本输入输入模式:

运行:e dapping.txt

按i键

输入下面这段文字(其中包含了很多错字和错误,以后我们会纠正):

means being determined about being determined and being passionate  
about being passionate

按<Esc>键回到normal mode (命令模式)。

运行:w

哎哟,我们似乎漏了一个单词,在行的起始位置,可我们的光标在行尾。我们现在该怎么做呢?用哪个最有效的方法我们才能回到行首,补上漏掉的单词?我们应该用鼠标光标移动到行首?我们应该用方向键慢慢的退回到行首?还是我们先用Home键回到行首,然后按i键再次回到插入模式?

已经证明最有效的方法是按I(大写键I):

按I键 (shift-i)

输入Dappin

按<Esc> 键,回到命令模式

注意到没有,我们用一个不同的键回到编辑模式,尤其是移动光标到了行首且切换到编辑模式。还有就是,一完成输入就回到命令模式很重要。把这作为一个好的习惯很有利,因为绝大多数的工作(在刚开始的输入阶段)都是在命令模式完成的--这就是所有重要的改写/编辑/给文档润色等的地方。

现在,我们来用一个与i命令稍有区别的命令。说明一下,按i键会把光标放在当前位置之前,切换到编辑模式。为了把光标放在当前位置 ("a"fter)之后,就要用a键。

按a键

输入g(彻底完成输入"Dapping")

按<Esc>键回到命令模式

---

与i键和I键的关系相似，在a键和A键之间也有一个关系- 假如你你想在行尾追加输入，就按A键。

按A键

输入。（输入一个句号，刚好完成整句）

按<Esc>键切换到命令模式。

总结一下目前为止我们学习的四个命令：

命令	运行
i	在光标前插入文本
I	在行首插入文本
a	在光标后追加文本
A	在行尾追加文本

注意，大写字母命令就是小写字母命令的‘bigger（功能更大）’版本。

现在我们已经非常熟悉了光标在一行内的速移功能，那我们就看看怎么速移到新行。假如你想开始("o"pen)编辑新行，就按o键。

按o键

输入I'm a rapper.

按<Esc>键切换到命令模式。

想一下，要是我们输入的这一句成为一段话，o命令的作用就更吸引人了。

按O键（大写 'O'）

Press<Esc>键回到道明令模式。

总结一下我们刚才学的两个新命令：

运行 结果

o 在当前行下打开一新行

O 在当前行上打开一新行

要注意的是大写和小写'o'命令在打开新行的时候方向相反。

刚才我们写的，有错误？哈，对，应该是'dapper'，不是'rapper'！就是一个字符，有最快的方法来改吗？我们用i键切换到编辑模式，按 <Del>键删除r，输入d，然后按<Esc>键回到编辑模式。

但是这样需要四步来操作！没有根快的了吗？用s键吧- s的意思就是替换（'s 'ubstitute）。

把光标移动到r字符上(或简单的按b键回到单词的起始位置)

按s键

输入d

按<Esc>键回到命令模式

好了，或许此刻这样的操作不会为我们节省多少时间，但是一天下来，一次一次的重复这样的事还是有效率的！这么普通的工作是越快越好，因为它帮助我们集中精力做更多有创意有性趣的事情。就如Linus Torvalds (Linux创始人)说的，"它不只是完成的更快，而且就因为如此的快，你的工作才得到了明显的改观。"

s键也有一个大写版本命令,S键替换整行的字符。

按S键

输入Be a sinner.

按<Esc>键 回到命令模式。

命令	结果
----	----

s	替换当前一个字符
---	----------

S	替换当前行所有字符
---	-----------

让我们回到上一节说的那个例子... 难道我们不能用'r'ep lace命令替换单个字符呢?是的，我们可以用r命令。

把光标移动到错误的字符上。

按r键

输入字符d

注意我们已经在命令模式下了，不需要再按<Esc>键了。

有个大写版本的R命令，它能替换连续的字符。移动光标到错误字符'i'上，按R键 输入app(这个单词就变成了'dapper')按<Esc>键回到命令模式。

---

命令	结果
r	替换当前字符
R	替换连续的字符

这段内容看起来应该像这样:

Dapping means being determined about being determined and being passionate about being passionate. Be a dapper.

好了,这一章我们说了很多,但我敢保证这是最难的一步。一旦你理解了,你就更加理解了Vim运行机制的重心和灵魂,并且Vim中的所有其他功能,只是附加上去的一些命令而已。重复一下,理解了模式的工作机制和模式之间切换工作是成为Vimmer的关键,因此,你没有必要再整理这章的内容,请你空闲的时候阅读一下,花点时间吧。

假如你想了解更多得详细内容,请运行:help inserting和:help replacing来查看。

## 可视模式

假如你想选择一块文本区域,用你所想的一段文本来完整的替换,要怎么做呢?

一个办法是用鼠标点击你要更换的文本区域,按住鼠标左键,拉动鼠标到相应文本的结尾,然后松开左键。说这些似乎跑题太远了。

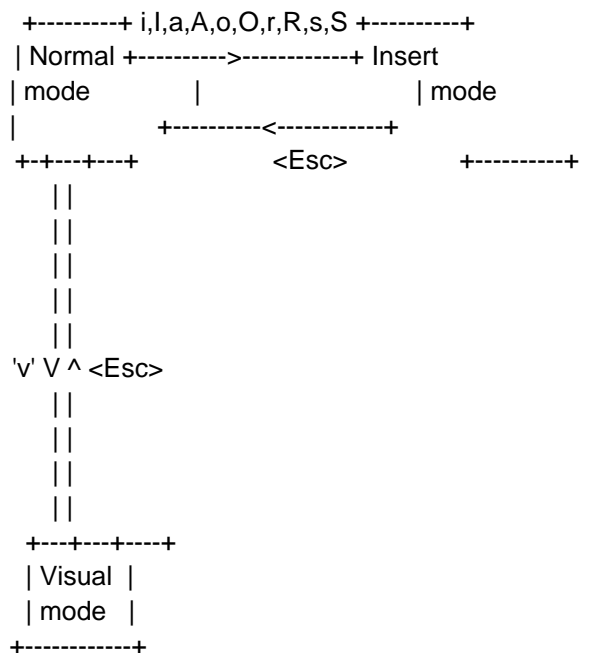
我们可以用<Del>键或者<Backspace>键删除所有的字符,但是效率似乎很低。

最有效率的方法是把光标移到文本的开头,按v键启动可视模式,用方向键或者任何一个移动命令移到相关文本的结尾(比如,按5e两键,从当前光标处移动到第五个单词上)然后按c键改变('c') hange 文本。看,效率得到了改进。

在这次操作中(c命令),完成后你将进入编辑模式,要是按<Esc>键的话将返回到命令模式。

## 总结

这里,画了一个不同模式之间的关系图:



这里画的，用的是Vim and Dr.Chip's DrawIt 插件[1])

请运行: `help vim-modes-intro`和:`help mode-switching` , 查看不同模式的细节和模式之间的转化内容. 如果你仍保持着模式概念在Vim强大功能和简易性的中心位置的疑惑, 只有请你阅读"Why Vi" [2]的相关章节和vi input model (输入模式) [3]的"why it is a better way of editing" (为什么它是一种好的编辑方法) 章节内容。

[前页](#)   [后页](#)

外部链接：

[1] [http://vim.sourceforge.net/scripts/script.php?script\\_id=40](http://vim.sourceforge.net/scripts/script.php?script_id=40)

[2] <http://www.viemu.com/a-why-vi-vim.html>

[3] <http://blog.ngedit.com/2005/06/03/the-vi-input-model/>

内容来自于: [http://www.swaroopch.com/mediawiki/index.php?title=Vim\\_en:Modes&oldid=1023](http://www.swaroopch.com/mediawiki/index.php?title=Vim_en:Modes&oldid=1023)

首席作者: Swaroop