技巧：快速提高 Vi/Vim 使用效率的原则与途径

Vi/Vim 是所有 Unix/Linux 操作系统默认配备的编辑器。因其强大的功能和高效的操作，Vi/Vim 也成为众多 Unix/Linux 用户、管理员必须掌握并熟练使用的编辑工具之一。尤其是在没有图形界面的情况下，更是离不开 Vi/Vim。Vi/Vim 命令非常多、用法极为灵活，掌握起来有一定的难度。本文作者将结合自己的使用经验，分七个方面着重介绍哪些原则、途径或命令能快速提高 Vi/Vim 文件编辑效率，从而达到事半功倍的效果。

方 吾松, 高级软件工程师, IBM
2012 年 4 月 19 日

+ 内容

5 评论

引言

Vi 最初是由 Bill Joy 在 1976 年编码实现的。而 Vim (Vi IMproved) 则是改进的 Vi，由 Bram Moolenaar 在 1991 年开发并发布。Vi/Vim 是所有 Unix/Linux 操作系统默认配备的编辑器。因其强大的功能和高效的操作，Vi/Vim 也成为众多 Unix/Linux 用户、管理员必须掌握并熟练使用的编辑工具之一。尤其是在没有图形界面的情况下，更是离不开 Vi/Vim。Vi/Vim 命令非常多、用法极为灵活，掌握起来有一定的难度。详细介绍 Vi/Vim 使用方法的文章有很多，因而本文就不一一述及 Vi/Vim 的诸多功能及命令。本文作者将结合自己的使用经验，分七个方面着重介绍哪些原则、途径或命令能快速提高 Vi/Vim 文件编辑效率，从而达到事半功倍的效果。本文述及的途径或命令均只针对系统默认配置的 Vi/Vim。关于 Vi/Vim 各种定制后的功能不在本文讨论范围之列。（**注：**本文中提到的所有命令均在 Red Hat Enterprise Linux Server release 6.1 上测试通过。）

↑ 回页首

Vi/Vim 版本的选择

“工欲善其事，必先利其器”。在 Vi/Vim 版本的选择上，原则是“**能用 Vim 就不要使用 Vi；能用最新版就不要守着旧版本**”。Vim 提供的功能和特性要比 Vi 多得多，如语法加亮着色功能等。就使用效果及效率来说，编辑同样的文件，使用 Vim 更胜一筹；就版本来说，新版的往往会修复旧版的一些缺陷及不足。这就要求我们在可能的情况下一定要使用最新版的 Vim。

小技巧：

在 Linux 下，如果以 root 用户登录系统的话，通过 vi 命令打开的 Vim 编辑器往往只加载最基本的功能，像语法加亮着色的功能基本上没有。在 root 用户下使用 Vim 所有功能的技巧是用 vim 命令打开 Vim 编辑器。

↑ 回页首

启动及关闭 Vi/Vim

打开及关闭 Vi/Vim 的方法有很多，既可以只启动 Vi/Vim 编辑器本身，也可以在启动 Vi/Vim 编辑器的同时打开一个或多个文件；既可以放弃存盘退出，也可以只保存文件的一部分。以下为相关命令列表：

表 1. 启动及关闭 Vi/Vim 的基本命令

功能	命令	说明
启动	vi 打开 Vi/Vim vi <file> 打开 Vi/Vim 并加载文件 <file>	
退出	ZQ 无条件退出 :q! 无条件退出 ZZ 存盘并退出 :wq 存盘并退出	
保存部分文件	:m,nw <file> 将 m 行到 n 行部分的内容保存到文件 <file> 中 :m,nw >> <file> 将 m 行到 n 行的内容添加到文件 <file> 的末尾	

掌握并熟练这些基本命令是使用 Vi/Vim 的基本要求。只有这样，才能在实际使用过程中做到按需选择，灵活使用，提高 Vi/Vim 的使用效率。需要说明的一点是：在使用 Vi/Vim 的时候，有时想临时退出 Vi/Vim，转到 shell 环境里去做一些操作，等这些操作结束后，再继续回到刚才的 Vi/Vim 状态。针对这一经常碰到的现实需求，很多人在大多数情况下会将保存退出 Vi/Vim，进入 shell 执行操作，然后再重新打开刚才编辑的文件。其实这是一个很低效的方法，因为再次打开需要重新定位刚才编辑的地方，麻烦不说，Vi/Vim 编辑器的状态也完全不一样了。其实，有两种方法可以可以实现临时退出 Vi/Vim、进入 shell 环境后再回来的要求：

方法一：使用 Ctrl-z 以及 fg 这两个命令组合。

这一解决方法主要利用了 Linux/Unix 的作业机制。具体原理是：Ctrl-z 命令将当前的 Vi/Vim 进程放到后台执行，之后 shell 环境即可为你所用；fg 命令则将位于后台的 Vi/Vim 进程放到前台执行，这样我们就再次进入 Vi/Vim 操作界面并恢复到原先的编辑状态。

方法二：使用行命令 :sh。

在 Vi/Vim 的正常模式下输入 **:sh**即可进入 Linux/Unix shell 环境。在要返回到 Vi/Vim 编辑环境时，输入 exit 命令即可。

这两种方法实现机制不一定，但效果一样，都非常快捷有效。

↑ 回页首

移动光标

Vi/Vim 中关于光标移动的命令非常多，这也是很多人经常困惑并且命令用不好的地方之一。其实 Vi/Vim 中很多命令是针对不同的操作单位而设的，不同的命令对应不同的操作单位。因而，在使用命令进行操作的时候，首先要搞清楚的就是要采用哪种操作单位，也就是说，是要操作一个字符，一个句子，一个段落，还是要操作一行，一屏、一页。单位不同，命令也就不同。只要单位选用得当，命令自然就恰当，操作也自然迅速高效；否则，只能是费时费力。这也可以说是最能体现 Vi/Vim 优越于其它编辑器的地方之一，也是 Vi/Vim 有人爱有人恨的地方之一。在操作单位确定之后，才是操作次数，即确定命令重复执行的次数。要正确高效的运用 Vi/Vim 的各种操作，一定要把握这一原则：**先定单位再定量**。操作对象的范围计算公式为：操作范围 = 操作次数 * 操作单位。比如：5h 命令左移 5 个字符，8w 命令右移 8 个单词。

注：有些操作单位（如文件）是不能加操作次数。具体说明请参考 Vi/Vim 使用手册。

Vi/Vim 中操作单位有很多，按从小到大的顺序为（括号内为相应的操作命令）：字符（h、l）→ 单词 (w、W、b、B、e) → 行 (j、k、0、^、$、:n) → 句子（(、)）→ 段落（{、}）→ 屏 (H、M、L) → 页（Ctrl-f、Ctrl-b、Ctrl-u、Ctrl-d) → 文件（G、gg、:0、:$）。

具体命令解释如下：

表 2. 移动光标的基本命令

操作单位	命令	说明
字符	h 左移一字符 l 右移一字符	
单词	w/W 移动到下一单词的开头 b/B 移动到上一单词的开头 e/E 移动到光标所在单词的末尾	W、B、E 命令操作的单词是以空白字符（空格、Tab）分隔的字串，比如字符串“str1-str2 str3-str4”，对 W、B、E 命令来说是两个单词，而对 w、b、e 命令来说则是四个单词。
行	j 下移一行 k 上移一行 0 移到当前行开头 ^ 移到当前行的第一个非空字符 $ 移到当前行末尾 :n 移动到第 n 行	0 为数字零（zero）
句子	) 移动到当前句子的末尾 (移动到当前句子的开头	
段落	} 移动当前段落的末尾 { 移到当前段落的开头	
屏	H 移动到屏幕的第一行 M 移动到屏幕的中间一行 L 移动到屏幕的最后一行	
页	Ctrl-f 向前滚动一页 Ctrl-b 向后滚动一页 Ctrl-u 向前滚动半页 Ctrl-d 向后滚动半页	
文件	G 移动到文件末尾 gg 移动到文件开头 :0 移动到文件第一行 :$ 移动到文件最后一行	0 为数字零（zero）

除了这些基本单位之外，还有 %（跳转到与之匹配的括号处），`.（跳转到最近修改过的位置并定位编辑点），'.（跳转到最近修改过的位置但不定位编辑点）这三个命令也非常重要，在 Vi/Vim 中灵活使用会极大提高效率。% 除用于光标移动之后，还可用于检测源码中各种括号的匹配情况。

↑ 回页首

文本编辑

与光标移动一样，Vi/Vim 中关于编辑操作的命令也比较多，但操作单位要比移动光标少得多。按从小到大的顺序为（括号内为相应的操作命令）：字符 （x、c、s、r、i、a）→ 单词 (cw、cW、cb、cB、dw、dW、db、dB) → 行 (dd、d0、d$、I、A、o、O) → 句子（(、)）→ 段落（{、}）。这些操作单位有些可以加操作次数。操作对象的范围计算公式为：操作范围 = 操作次数 * 操作单位。比如：d3w 命令删除三个单词，10dd 命令删除十行。

具体命令解释如下：

表 3. 文本编辑的基本命令

操作单位	命令	说明
字符	x 删除光标位置的字符 c 更改当前字符并进入插入模式 s 替换光标位置的字符并进入插入模式 r 替换光标位置的字符但不进入插入模式 i 在当前位置的字符之前进入插入模式 a 在当前位置的字符之后进入插入模式	
单词	cw/cW 删除当前单词从光标开始的部分并进入插入模式 cb/cB 删除当前单词从光标所在位置至单词开始的部分并进入插入模式 dw/dW 删除当前单词从光标开始的部分但不进入插入模式 db/dB 删除当前单词从光标所在位置至单词开始的部分但不进入插入模式	cW、cB、dW、dB 命令操作的单词是以空白字符（空格、Tab）分隔的字串，比如字符串“str1-str2 str3-str4”，对 cW,cB,dW,dB 命令来说是两个单词，而对 cw、cb、dw、db 命令来说则是四个单词。
行	dd 删除当前行 d0 删除从当前光标开始到行末的内容 d$ 删除从当前光标开始到行末的内容 I 在当前行的行首进入插入模式 A 在当前行的行尾进入插入模式 o 在当前行下方另起一行进入插入模式 O 在当前行上方另起一行进入插入模式	▪ d0 命令中的 0 为数字零 ▪ o 为小写英文字母 [əu] ▪ O 为大写英文字母 [əu]
句子	d) 删除当前句子从光标位置开始到句末的内容 d(删除当前句子从光标位置开始到句首的内容	
段落	d} 删除当前段落从光标位置开始到段末的内容 d{ 删除当前段落从光标位置开始到段首的内容	

除上述最基本的文本编辑命令这外，Vi/Vim 还提供了许多其它的编辑命令或相关组合。使用这些命令或相关组合往往在极大提高文本编辑的效率与速度。现将这些命令按功能列如表下：

表 4. 文本编辑的高效命令

功能	命令	说明
复制与粘贴	yw 复制当前单词从光标开始的部分 yy 复制光标所在行的所有字符 p 将最后一个删除或复制文本放在当前字符 P 将最后一个删除或复制文本放在当前字符之前	配合操作数使用可快速拷贝编辑文本
撤消与重做	u 撤消更改 Ctrl-R 重做更改	非常实用的一个命令
重复操作	. 重复上次操作	▪ . 为小数点（dot） ▪ 配合光标移动命令使用； ▪ 不用重复输入先前的复杂命令即可在不同的地方做同样的操作，有点象 MS Office 的格式刷
交换相邻字符或行	xp 交换光标位置的字符和它右边的字符 ddp 交换光标位置的行和它的下一行	
大小写转换	~ 将光标下的字母大小写反向转换 guw 将光标所在的单词变为小写 guw 将光标所在的单词变为小写 gUw 将光标所在的单词变为大写 guu 光标所在的行所有字符变为小写 gUU 光标所在的行所有字符变为大写 g~~ 光标所在的行所有字符大小写反向转换	
取得外部输入	: r!<cmd> 将命令 <cmd> 的输出结果插入到当前光标所在位置 :r <file> 将文件 <file> 读入到当前光标所在位置	
排序	:1,$!sort 将文件内的所有内容排序	
加入行号	:%!nl 在所有非空行前加入行号 :%!nl -ba 在所有行前加入行号	利用 Linux 命令 nl 来实现的
缩进	>> 右缩进（可配合操作数使用） << 左缩进（可配合操作数使用）	配合操作数使用，在编辑源码的时候非常有用。
自动补全	Ctrl-p 自动补全	在编写代码的时候非常有用。比如，输入 prin 后按 Ctrl-p 将自动帮你输入 printf 函数名后面的部分，同时将相关备选函数在底下列出来。
显示当前编辑文件名	Ctrl-g 显示当前编辑文件名及行数	可以在不退出 Vi/Vim 情况下了解当前编辑文件的信息
显示字符内码	ga 显示光标所在字符的内码（包括十进制码，十六进制码以及八进制码）	显示的内码为当前 encoding 下的内码

↑ 回页首

文本搜索与替换

Vi/Vim 用于文本搜索的主要有下面的三个基本命令：

表 5. 文本搜索的基本命令

功能	命令	说明
搜索	/ 在文件中向前搜索 ? 在文件中向后搜索	▪ 可使用正则表达式 ▪ 可配合操作数使用，比如 3/str 向前搜索字串 str 并将光标移到第三个找到的串
搜索下一个	n 搜索下一个 N 反向搜索下一个	▪ 可使用正则表达式 ▪ 可配合操作数使用，比如命令 3N 反向搜索第三个匹配的字符串

除这三个基本命令之外，还有以下三个非常有效快捷的与搜索查找有关的命令：

表 6. 文本搜索的高效命令

功能	命令	说明
快速搜索	* 在文件中向前搜索当前光标所在的单词 # 在文件中向后搜索当前光标所在的单词	非常快捷的搜索命令
显示搜索命令历史	q/ 显示搜索命令历史的窗口 q? 显示搜索命令历史的窗口	可以选择重用以前用过的搜索查找命令
查找帮助	Shift-k 查找光标所在命令或函数的 man 帮助	▪ 可以在不退出 Vi/Vim 情况下快速查询命令或函数的使用方法； ▪ 按 q 键退出 man 帮助

关于替换主是要结合搜索使用行命令来实现，命令格式为：

:m,ns/str1/str2/g 将 m 行到 n 行中的字串 str1 全部替换为字串 str2。

在众多使用正则表达式进行替换的命令中，平时需要掌握的一个命令是如何快速去除 ^M 字符。在 Linux/Unix 系统中编辑 Windows 操作系统中生成的文件时，如果上传时回车换行符处理不正确的话，用 Vi/Vim 打开经常会出现 ^M 字符。如果上传的文件是 shell 脚本的话，即使赋予了正确的执行权限该脚本还是无法运行。这也是各种文件在 Linux/Unix 和 Windows 中传输经常会出现问题的地方之一。其实，在 Vi/Vim 中使用替换命令 :**1,$s/^M//g**即可以很容易地快速去掉 ^M 字符。这里需要注意的是 ^M 是使用 Ctrl-v、Ctrl-m 输入的。

↑ 回页首

执行外部命令

在 Vi/Vim 中还可以在正常模式下执行各种外部命令，命令格式如下：

表 7. 执行外部命令格式

功能	命令	说明
执行外部命令	:!<cmd> 执行外部命令 <cmd>	在正常模式下输入该命令
显示命令行命令历史	q: 显示命令行命令历史的窗口	可以选择重用以前用过的命令行命令

其中命令 **q:**会显示使用过的行命令历史，可以从中选择重用以前用过的命令。这对于需要重复应用那些复杂的命令来说，非常方便快捷。

↑ 回页首

Vi/Vim 设置

Vi/Vim 有很多内部变量，可以根据需要进行相应的设置。变量类型不同往往设置方式也不一样，简单的只要设置特定的变量名即可，复杂的则需要指定和分配一个显式值来设置变量。在实际应用中，如果有需要，请参考 Vi/Vim 的使用手册。这里主要列出大家经常使用并能提高编辑效率的一些设置命令：

表 8. Vi/Vim 设置命令

功能	命令	说明
查看设置的当前值	:set all 查看 vi 或 Vim 中设置的所有选项的当前值 :set <option>? 查看特定选项 <option> 的当前值	
设置行号显示与否	:set number 显示行号 :set no number 取消行号显示	命令的简写形式： :set nu :set no nu
设置自动缩进	:set autoindent 设置自动缩进 :set no autoindent 取消自动缩进设置	命令的简写形式： :set ai :set no ai
设置缩进宽度	:set shiftwidth=4 设置缩进宽度为 4	命令的简写形式： :set sw=4
设置大小写忽略与否	:set ignorecase 设置忽略大小 :set no ignorecase 取消忽略大小设置	命令的简写形式： :set ic :set no ic
设置不可见字符显示与否	:set list 显示不可见字符 :set nolist 取消显示不可见字符设置	在显示不可见字符的情况下，TAB 键显示为 ^I，而 $ 显示在每行的结尾。

在正常模式 (Normal mode) 时，执行这些设置命令只修改当前会话的设置，退出或重启 Vi/Vim 这些设置就丢失了。要想保持住这些设置，就必须将这些设置写入 Vi/Vim 的 vimrc 文件。对 Linux/Unix 操作系统来说，每个用户的 vimrc 文件位于该用户的主目录下，文件名为 .vimrc。Vi/Vim 在每次启动的时候都会读取用户主目录下的 vimrc 文件并据此设置 Vi/Vim 的使用环境。Vi/Vim 在安装的同时也会安装 vimrc 文件的一个示例 vimrc_example.vim 到 /usr/share/vim/vim<version> 目录下。可以根据需要将这个示例文件拷贝到当前用户的主目录下并重命名为 .vimrc，在此基础上进行修改会相对容易一些。其它关于 Vi/Vim 的一些定制和键映射等相关设置基本上也是写入用户的 vimrc 文件中，更详细的介绍请参考 Vi/Vim 的使用手册。用户也可以维护一个自己的 vimrc 文件，并将这个文件拷贝到自己的使用的环境中，保持不同环境中 Vi/Vim 特性的一致，以符合自己的使用习惯。

↑ 回页首

结束语

本文主要介绍了快速提高 Vi/Vim 使用效率的途径及相关命令，熟练掌握这些是高效使用 Vi/Vim 的基本要求。关于 Vi/Vim，还有很多其它命令以及定制功能本文没有涉及到。大家可结合自己的使用情况及需求，在本文的基础上进行拓展，不断提高自己的 Vi/Vim 使用水平。

参考资料

学习

- 访问 developerWorks 技术专题 AIX and UNIX，阅读 AIX/Unix/Linux 系统管理以及相关软件应用方面的文章，扩展自己的 AIX/Unix/Linux 技能。
- 查看文章“Vim 实用技术，第 1 部分: 实用技巧”，了解 Vim 的五个操作模式。
- 查看文章“对话 UNIX: 新改进的 Vim 编辑器”，了解 Vi 与 Vim 之间的差异及 Vimr 基本命令。
- 查看文章“神奇的 VIM”，了解 Vim 强大的文本处理功能及相关命令或脚本。
- 查看文章“vi 技巧和诀窍：令人刮目相看的 10 个超酷命令”，了解 Vim 的使用技巧及诀窍。
- 查看文章“学习 Linux，101: 使用 vi 编辑文件”，了解 Vi 的基本使用方法。
- 在 developerWorks Linux 专区 寻找为 Linux 开发人员（包括 Linux 新手入门）准备的更多参考资料，查阅我们 最受欢迎的文章和教程。
- 在 developerWorks 上查阅所有 Linux 技巧 和 Linux 教程。
- 随时关注 developerWorks 技术活动和网络广播。

讨论