Coder

商业化产品可能是有用的， 但最好的东西往往是免费的， 空气，水，Vim。

vim(二) Global command

2013-04-10 21:341871 人阅读评论(0)收藏举报

分类：

vim（21）

发信人: vale (浅谷), 信区: VIM

标 题: global 命令详解

发信站: 水木社区 (Fri Jun 15 17:05:55 2007), 站内

global 命令是 Vim 最强大的命令之一（个人认为是 No.1），将其摸透用熟可以事半功倍，

本文总结了版上的一些经典问题，结合自己的使用和理解，试图通过实例详细

介绍一下

其用法。示例难度不一，有些并没有多少实用性，为题而生，读者各取所需吧。

示例说

明并不非常细致，以免罗唆。每节标题下列出了所涉及的内容在 Vim help 中的位置，以

供查找。文中用词未必标准（我没看过 Vim 中文帮助），观点也难免有错，请大家指正！

1. global 命令形式

:h :g

:h 12.4

:[range]global/{pattern}/{command}

global 命令在[range]指定的文本范围内（缺省为整个文件）查找{pattern}，然后对匹

配到的行执行命令{command}，如果希望对没匹配上的行执行命令，则使用 global!或 vg

lobal 命令。

先来看 Vim 用户手册里的一个经典例子。

【例 1】倒序文件行（即 unix 下的 tac 命令）

:g/^/m 0

这条命令用行首标记/^/匹配文件的所有行（这是查找的一个常用技巧，如果用 /./则是

匹配非空行，不满足本例要求），然后用 move 命令依次将每行移到第一行

（第 0 行的下一

行），从而实现了倒序功能。

global 命令实际上是分成两步执行：首先扫描[range]指定范围内的所有行，给匹配{pa

ttern}的行打上标记；然后依次对打有标记的行执行{command}命令，如果被标记的行在

对之前匹配行的命令操作中被删除、移动或合并，则其标记自动消失，而不对该行执行

{command}命令。标记的概念很重要，以例说明。

【例 2】删除偶数行

:g/^/+1 d

这条命令也是匹配所有行，然后隔行删除（其中+1 用以定位于当前行的下一行）。为什

么是隔行呢？因为在对第一行执行+1 d 命令时删除的是第二行，而第二行虽然也被标记

了，但已不存在了，因此不会执行删除第三行的命令。

本例也可以用 normal 命令实现：

:%norm jdd

%指定整个文件，然后依次执行普通模式下的 jdd，即下移删除一行。与 global 命令不同

之处在于，%norm 是按照行号顺序执行，在第一行时删除了第二行，后面的所有行号都减

一，因此在第二行执行 jdd 时删除的是原来的第四行。也就是说，global 命令是通过偶数

行标记的消失实现的，而 normal 命令是通过后续行的自动前移实现的。

【例 3】删除奇数行

:g/^/d|m.

光是:g/^/d 显然不行，这会删除所有行，我们需要用 move 命令把偶数行的标记去掉。当

然，本例可以很简单的转换成【例 2】，在此只是用来强调标记的概念。

本例若想用 normal 命令实现比较有意思，%norm dd 也同样会删除整个文件，%norm jkdd

就可以，我不知道两者为什么不同，可能和 normal 命令内部的运行机制有关。

2. global 与 substitute

:h 10.4

:helpg ms-word\c

不少 vimmer 觉得这两个命令差不多，的确，它们的形式很相似，都是要进行查找匹配，

只不过 substitute 执行的是替换而 global 执行的其它命令（当然，substitute 缺省的[r

ange]是当前行，这点也不同）。先看两个例子，体会一下:s 和:g 不同的思维方式。

【例 4】double 所有行

:%s/.*/&\r&/

:g/^/t.

substitue 是查找任意行，然后替换为两行夹回车；global 是将每一行复制（:t 就是:co

py）到自己下面，更加清晰明了。

【例 5】把以回车排版、以空行分段的文本变成以回车分段的文本

很多 txt 格式的 ebook，以及像 vim help 这样的文本，每行的字符数受限，段之间用空行

分隔。若把它们拷贝到 word 里，那些硬回车和空行就比较讨厌了，虽然 word 里也有自动

调整格式的功能，不过在 Vim 里搞定更是小菜一碟。先看看用替换如何实现。

:s/\n\n\@!/ /

\n\n\@!是查找后面不跟回车的回车（关于\@!的用法请:h /\@!，在此不多说了），然后

替换为空格，也就是去掉用于排版的回车。global 命令则完全是另一种思路。

:g/./,/^$/j

/./标记非空行，/^$/查找其后的空行，然后对二者之间的行进行合并操作。也许有人会

问，段中的每一行会不会都执行了 j 命令？前面已经说过，在之前操作中消失掉的标记行

不执行操作命令，在处理每段第一行时已经把段内的其余行都合并了，所以每段只会执

行一次 j 命令。这条命令使用 global 标记做为[range]的起始行，这样的用法

后面还会详

述。

global 经常与 substitute 组合使用，用前者定位满足一定条件的行，用后者在这些行中

进行查找替换。如：

【例 6】将 aaa 替换成 bbb，除非该行中有 ccc 或者 ddd

```
:v/ccc\|ddd/s/aaa/bbb/g
```

【例 7】将 aaa 替换成 bbb，条件是该行中有 ccc 但不能有 ddd

如何写出一个匹配 aaa 并满足行内有 ccc 但不能有 ddd 的正则表达式？我不知道。即便能写

出来，也必定极其复杂。用 global 命令则并不困难：

```
:g/ccc/if getline('.') !~ 'ddd' | s/aaa/bbb/g
```

该命令首先标记匹配 ccc 的行，然后执行 if 命令（if 也是 ex 命令！），

getline 函数取得

当前行，然后判断是否匹配 ddd，如果不匹配（!~的求值为 true）则执行替换。要掌握这

样的用法需要对 ex 命令、Vim 函数和表达式有一定了解才行，实际上，这条命令已经是一

个快捷版的脚本了。可能有人会想，把 g 和 v 连起来用不就行了么，可惜 global 命令不支

持（恐怕也没法支持）嵌套。

3. global 标志的[range]用法

:h range

在 global 命令第一步中所设的标记，可以被用来为{command}命令设定各种形式的[rang

e]。在【例 2】和【例 5】中都已使用了这一技巧，灵活使用[range]，是一项重要的基本

功。先看看【例 2】和【例 3】的一般化问题。

【例 8】每 n 行中，删除前/后 m 行（例如，每 10 行删除前/后 3 行）

```
:g/^/,+2 d | ,+6 m -1
:g/^/,+6 m -1 | +1,+3 d
```

这两个命令还是利用 move 来清除保留行的标志，需要注意的是执行第二个命令时的当前

行是第一个命令寻址并执行后的位置。再看两个更实用点的例子。

【例 9】提取条件编译内容。例如，在一个多平台的 C 程序里有大量的条件编译代码：

```
#ifdef WIN32
  XXX1
  XXX2
#endif
 ...
#ifdef WIN32
  XXX3
  XXX4
```

```
#else
    YYY1
    YYY2
#endif
```

现在用 global 命令把 Win32 平台下代码提取出来，拷贝到文件末：

```
:g/#ifdef WIN32/+1,/#else\|#endif/-1 t $
```

t 命令的[range]是由逗号分隔，起始行是/#ifdef WIN32/标记行的下一行，结束行是一
个查找定位，是在起始行后面出现的#endif 或#else 的上一行，t 将二者间的内容复制到
末尾。

【例 10】提取上述 C 程序中的非 Win32 平台的代码（YYY 部分）

首先说明一下，这个例子比前例要复杂的多，主要涉及的是[range]的操作，已经和 glo
bal 命令没多少关系，大可不看。加到这的目的是把问题说完，供喜欢细抠的朋友参考。

本例的复杂性在于：首先，不能简单的用#else 和#endif 定位，因为代码中可能有其它的
条件编译，我们必须要将查找范围限定在#ifdef WIN32 的 block 中；另外，在 block 中可
能并没有#else 部分，这会给定位带来很大麻烦。解决方法是：

```
:try | g/#ifdef WIN32//#else/+1, /#endif/-1 t $ | endtry
```

先不管 try 和 endtry，只看中间的 global 部分：找到 WIN32，再向后找到 #else，将其下一
行作为[range]的起始行，然后从当前的光标（WIN32 所在行，而非刚找到的 #else 的下一
行）向下找到#endif，将其上一行作为[range]的结束行，然后执行 t 命令。但对于没有
#else 的 block，如第一段代码，[range]的起始行是 YYY1，而结束行是 XXX2（因为查找#
endif 时是从第一行开始的，而不是从 YYY1 开始），这是一个非法的[range]，会引起 ex
ception，如果不放在 try 里面 global 命令就会立刻停止。
与逗号(,)不同，如果[range]是用分号(;)分隔的，则会使得当前光标移至起始行，在查
找#endif 时是从#else 的下一行开始，就不会产生非法[range]，用不着 try，但带来的问
题是：没有#else 的 block 会错误的把后面 block 中的#else 部分找出来。
4. global 与 Vim 脚本
:h script
:h expression
经常有人问：XxEditor 有个什么功能，Vim 支持么？很可能不支持，因为 Vim 不大会为特
定用户群提供非一般化的功能，但很少有什么功能不能在 Vim 定制出来，如果

是你常用，

就加到你的 vimrc 或者 plugin 里。脚本就是定制 Vim 的一种利器。本文不讨论脚本的编写

，而是介绍如何实用 global 实现类似脚本的功能，实际上，就是利用命令提供的机制，

做一个简化的脚本。

【例 11】计算文件中数字列之和（或其它运算）

```
:let i=0
:g/^/let i+=str2nr(getline('.'))
:echo i
```

首先定义变量 i 并清零，然后用 str2nr 函数把当前行转成数字累加到 i 中，注意 Vim 不支持

浮点数。global 在这里实际上是替代了脚本里的 for 循环。

Vim 中最常见的一个问题是如何产生一列递增数字，有很多解决办法，调用外部命令，录

宏，用 substitute 命令，还有专门的插件，而用 global 命令，可以实现一些更高级的功

能。见下例。

【例 12】给有效代码行添加标号

在_Data Structures and Algorithm Analysis in C_一书中，作者为了便于讨论，将代

码中的有效行加上注释标号，例如：

```
/*  1 */  unsigned int factorial( unsigned int n )
          {
/*  2 */      if( n <= 1 )
/*  3 */          return 1;
/*  4 */      return( n * factorial(n-1) );
          }
```

为了在添加标号后能对齐，我们预先在每行代码前面插入足够多的空格（这当然很简单

），然后用 global 命令自动添加标号：

```
:let i=1 | g/\a/s/ \{8}/\=printf("\/* %2d *\/",i)/ | let i+=1
```

其中变量 i 用来记录标号，g 命令查找有字母的行，然后把前 8 个空格替换成注释标号，每

行处理完成后标号加一。替换中用到了/\=，一个非常有用的功能。

最后我们再回到删除特定行的例子，用变量来搞定。

【例 13】每 n 行中，删除前/后 m 行

```
:let n=10 | let m=3
:let i=-1 | g/^/let i+=1 | if i%n
:let i=-1 | g/^/let i+=1 | if i%n>(n-m-1) | d
```

我们用 i 来记录处理行的位置，通过 i 的值与 m 和 n 的关系决定何时进行删除操作，这样的处

理方式比【例 8】的方法要清晰明了很多，而且更加通用。

5.小结

要用好 global 命令并非易事，命令中的每一部分都值得仔细研究：只有掌握了 range 原理
，才能自如的在文件中定位；只有精通 pattern，才能有效的匹配到想要找的行；只有熟
悉 ex 命令，才能选用最合适的功能进行操作；只有对变量、表达式、函数等内容有一定
了解，才更能让 global 命令实现脚本的功能。总之，global 是一个非常好的框架，对 Vi
m 越是熟悉，就越能将其种种武器架设在其上使用，发挥更大的威力。

global 当然并非万能，功能也有所欠缺，最主要的问题是只能用正则表达式来标志匹配
行，如果能用任意表达式来标记（或者从另一个角度，如前 mv 版主 runsnake 所说，引入
求值正则表达式），则可实现更加方便功能。比如前述的几个删除特定行的问题，将会
有简单而统一的解决方法。上述例子如果用 sed、awk 等专门的文本处理工具，或者 perl
之类的 script 语言也非难事，有些实现起来会更加方便。本文提供的 Vim 解决方法未必简
单，甚至可能是难于理解，目的在于介绍 global 的使用。对于那些不会或者不能使用其
它工具的朋友，参考价值可能更大一些。其实 Vim 的功能实在很丰富，值得我

们深入学习

。打个不恰当的比方，少林七十二绝技固然高妙，会的越多自然功力越强，不过只要会

上一门六脉神剑或小无相功，也足以独步江湖了。

14919176.05579

- vim(一) 入门
- vim(三) 正则匹配查找

我的同类文章

vim（21）

- •vim mark2015-04-29 阅读 314
- •ctags 使用细节 2014-04-03 阅读 506
- •vim:Increasing or decreasing numbers2013-11-27 阅读 647
- •vim 的使用好处（效率+多面手）2013-06-20 阅读 1215
- •vim 中单词拼写检查 spellchecking2013-06-12 阅读 2228
- •vim ctags 下 python 系统文件定义的跳转 2013-06-08 阅读 3849
- •Meet UltiSnips2014-07-04 阅读 511
- •vim(一)： 小技巧 2013-12-13 阅读 604
- •Running a Vim macro on a set of lines with norm2013-07-23 阅读

752

- •Vim 101: Search and Replace on Multiple Files2013-06-19 阅读 1185
- •vim map nmap...2013-06-09 阅读 631

更多文章

猜你在找

Windows Server 2012 DHCP Server 管理

Windows Server 2012 DHCP Server 管理

Part 23：Cocos2d-x 开发实战-移植-从 Win32 到 Windows Phone8

Windows Server 2012 Active Directory 管理

开发 LinuxShell 脚本程序【一】

vlc 分析

高通平台 android 环境配置编译及开发经验总结

VLC 框架分析

VLC 简介及使用说明

VLC 框架分析

关闭

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表 CSDN 网站的观点或立场

核心技术类目

全部主题 HadoopAWS 移动游戏 JavaAndroidiOSSwift 智能硬件

DockerOpenStackVPNSparkERPIE10EclipseCRMJavaScript 数据库

UbuntuNFCWAPjQueryBIHTML5SpringApache.NETAPIHTMLSDKIISFedo

raXMLLBSUnitySplashtopUMLcomponentsWindows

MobileRailsQEMUKDECassandraCloudStackFTCcoremailOPhone

CouchBase 云计算 iOS6Rackspace Web

AppSpringSideMaemoCompuware 大数据

aptechPerlTornadoRubyHibernateThinkPHPHBasePureSolrAngularCloud

FoundryRedisScalaDjangoBootstrap

wangeen's technique blog

个人资料

wangeen

- 访问：610459 次

- 积分：8302
- 等级：

- 排名：第 1988 名
- 原创：232 篇
- 转载：155 篇
- 译文：2 篇
- 评论：32 条

文章搜索

博客专栏

	MPI 分布式编程文章：10 篇 阅读：17754

	VIM 文章：21 篇 阅读：28319

	BOOST 文章：11 篇 阅读：13322

文章分类

- linux(40)
- mac(48)
- Libraries(9)
- Go language(2)
- C++ language(39)
- Tools(13)
- 并行计算(18)
- 应用(17)
- java(2)
- python(35)
- qt(9)
- boost(9)
- vim(22)
- git(9)
- web(17)
- windows(1)

- 开发方法(1)
- photoshop(3)
- bashrc(1)
- Effective STL(3)
- jquery(1)
- javascript(1)
- put(0)
- tkinter(1)
- twisted(1)
- GDB(4)
- django(11)
- 推荐系统(1)
- Aaron Swartz(1)
- web ios(1)
- websocket(1)
- sqlite(2)

文章存档

- 2015 年 11 月(1)
- 2015 年 10 月(3)
- 2015 年 06 月(1)
- 2015 年 04 月(1)
- 2015 年 02 月(4)

展开