|  |  |
| --- | --- |
| * **移动光标的方法** * **[Ctrl] + [f] 屏幕『向下』移动一页,相当于 [Page Down]按键 (常用)** * **[Ctrl] + [b] 屏幕『向上』移动一页,相当于 [Page Up] 按键 (常用)** * [Ctrl] + [d] 屏幕『向下』移动半页 * [Ctrl] + [u] 屏幕『向上』移动半页 * + 光标移动到非空格符的下一列 * - 光标移动到非空格符的上一列 * 20<space> 则光标会向后面移动 20 个字符距离。 * **0 或功能键[Home] 这是数字『 0 』 :移动到这一列的最前面字符处 (常用)** * **$ 或功能键[End] 移动到这一列的最后面字符处(常用)** * H 光标移动到这个屏幕的最上方那一列的第一个字符 * M 光标移动到这个屏幕的中央那一列的第一个字符 * L 光标移动到这个屏幕的最下方那一列的第一个字符 * **G 移动到这个文件的最后一列(常用)** * 20G 则会移动到这个文件的第 20 列(可 nG 配合 :set nu) * **gg 移动到这个文件的第一列,相当于 1G 啊! (常用)** * **n<Enter> n 为数字。光标向下移动 n 列(常用)**   删除、复制与贴上 | |
| * X 为向前删除一个字符(相当于 [backspace] (常用) * p 为将已复制的数据在光标下一列贴上 (常用) * dd 删除游标所在的那一整列(常用) * ndd删除光标所在的向下 n 列 (常用) * d1G 删除光标所在到第一列的所有数据 * yy 复制游标所在的那一列(常用) * nyy 复制光标所在的向下 n 列(常用) * y1G 复制光标所在列到第一列的所有数据 * J 将光标所在列与下一列的数据结合成同一列 * u 复原前一个动作。(常用) * [Ctrl]+r 重做上一个动作。(常用) | * x 为向后删除一个字符 (相当于 [del] 按键) (常用) 连续删除 10 个字符, 『10x』 * P 则为贴在游标上一列! (常用) * d$ 删除游标所在处,到该列的最后一个字符 * d0 删除游标所在处,到该列的最前面一个字符 * dG 删除光标所在到最后一列的所有数据 * y0 复制光标所在的那个字符到该列行首的所有数据 * y$ 复制光标所在的那个字符到该列行尾的所有数据 * yG 复制光标所在列到最后一列的所有数据 * c 重复删除多个数据,例如向下删除 10 列,[ 10cj ] * 『.』小数点是重复前一个动作的意思。 (常用) |
| * **搜寻与取代** * /word 向光标之下寻找一个名称为 word 的字符串。 (常用) * ?word 向光标之上寻找一个字符串名称为 word 的字符串。 * 使用 /word 配合 n 及 N 可以让你重复的找到一些你搜寻的关键词! * 『:100,200s/vbird/VBIRD/g』 在 100 到 200 列之间搜寻 vbird 并取代为VBIRD (常用) * :1,$s/word1/word2/g 从第一列到最后一列取代 word2 !(常用) * :1,$s/word1/word2/gc 取代前显示提示字符给用户确认 (confirm) (常用) * 指令列模式的储存、离开等指令 * :wq 储存后离开 :wq! 则为强制储存后离开 (常用) * ZZ 若文件没有更动,则不储存离开,若文件已经被更动过,则储存后离开! * :w [filename] 将编辑的数据储存成另一个文件(类似另存新档) * :r [filename] 将 『filename』 这个文件内容加到游标所在列后面 * :n1,n2 w [filename] 将 n1 到 n2 的内容储存成 filename 这个文件。 * 『:! ls /home』 在 vi 当中察看 /home 底下以 ls 输出的文件信息! * :set nu 显示行号 :set nonu 取消行号! * 区块选择的按键意义 * v 字符选择,会将光标经过的地方反白选择! * V 列选择,会将光标经过的列反白选择! * [Ctrl]+v 区块选择,可以用长方形的方式选择资料 * y 将反白的地方复制起来 * d 将反白的地方删除掉 * p 将刚刚复制的区块,在游标所在处贴上! * 多文件编辑的按键 * :n 编辑下一个文件 * :N 编辑上一个文件 * :files 列出目前这个 vim 的开启的所有文件 * 多窗口情况下的按键功能 * :sp [filename] 开启一个新窗口,如果有加 filename, 表示在新窗口开启一个新文件,否则表示两个窗口   为同一个文件内容(同步显示)。   * [ctrl]+w+↓(k) 按键的按法是:先按下 [ctrl] 不放, 再按下 w 后放开所有的按键,然后再按下 j (或向 * 下箭头键),则光标可移动到下方的窗口。 * [ctrl]+w+↑( j ) 同上,不过光标移动到上面的窗口。 * [ctrl]+w+ q <=> :q * 补齐的内容 * [ctrl]+x -> [ctrl]+n 透过目前正在编辑的这个『文件的内容文字』作为关键词,予以补齐 * [ctrl]+x -> [ctrl]+f 以当前目录内的『文件名』作为关键词,予以补齐 * [ctrl]+x -> [ctrl]+o 以扩展名作为语法补充,以 vim 内建的关键词,予以补齐      * Ctrl + x 数字减1 10 Ctrl + x 减10 * Ctrl + c 数字加1 * [esc] /word [esc] \* (或者直接把光标移到单词word上，再按\*) * 查找所有单词word, 按n 或 \* 下一个 : set hls 可高亮 * %s /word1/word2/ g word1替换成word2 * 把光标位于word1首， Cw word2 [esc] . n . word1 替换成 word2 * [ Insert ] 插入替换模式切换 R 进入替换模式 * V 进入可视模式 ctrl + g 可视 选择模式切换 * : ! {cmd} 执行一次shell命令 : ! python % [ % ] 表示当前文件 * : Ctrl + z 挂起vim进程， fg恢复进程， jobs查看被挂机的进程 * [ 替代: : shell 执行多次shell 命令， exit退出 ] * : %p 打印所有行 符号 % 代表当前文件中的所有行 * : <C-r><C-w> 用于插入光标下的单词, * : <C-r><C-a> 插入光标下的字串 | |
| * **创建快速遍历 Vim 列表的按键映射项**   用 :bn 和 :bp 来遍历缓冲区列表可以省点儿事,不过还是比较麻烦。为了  更快地遍历缓冲区列表,我采用了下面这些映射项,它们是在 Tim Pope 的  unimpaired.vim 插件中定义的:  Nnoremap <silent> [b :bprevious<CR>  Nnoremap <silent> ]b :bnext<CR>  Nnoremap <silent> [B :bfirst<CR>  Nnoremap <silent> ]B :blast<CR>  Vim 已经用 [ 和 ] 键作为一系列相关命令的前缀了,因此上面这些映射项  的风格与其一致。除上面这些之外, unimpaired.vim 插件还提供了其他一  些类似的映射项,分别用来遍历参数列表([a 和 ]a)、quickfix 列表( [q和 ]q)、  位置列表([l 和 ]l)以及标签列表([t 和 ]t )。   * **映射 Tab 键时需注意的事项**   Vim 把 <C-i> 和 <Tab> 当成同一个东西。如果你把 <Tab> 键映射成了其他  功能,那么也将会改变 <C-i> 命令的缺省行为。   * **Vim 术语对照表**   剪切(cut)、复制(copy)与粘贴(paste),Vim 使用的是另外的  术语 delete、yank 与 put。  Vim 中真正删除文本的操作是什么。也就是说,我们怎样才能  删除文本而不把其内容复制到任何寄存器?  答案是使用名为“黑洞”的特殊寄存器, "\_d{motion}执行真正的删除操作。  : set shiftwidth=4 softtabstop=4 expandtab 设置缩进为4个空格  : set history=200 设置历史命令记录为200条  : set hls 可高亮显示查找的词（\*）  set nrformats= 把所有数字到当成 10 进制  试着把下行内容加入你的 vimrc 文件:  cnoremap <expr> %% getcmdtype( ) == ':' ? expand('%:h').'/' : '%%'  现在,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就像你输入了 %:h<Tab>  从列块可视模式切换到插入模式: i 和 a 键会被当作一个文本对象的组成部分。。   * pwd [-P] :显示出确实的路径,而非使用链接 (link) 路径。   mkdir -p test1/test2/test3 [ -p ] 可以自行帮你建立多层目录  mkdir -m 711 test2 [ -m ] 权限设置  [  echo $PATH  PATH="${PATH}:/root"  ls -al --full-time ~  cp [options] source1 source2 source3 .... Directory  cp -a /var/log/wtmp . <==想要复制到当前目录,最后的 . 不要忘  ·[ -a ]整个资料特性完全一模一样  cp -r /etc/ /tmp  [ -r ] 是可以复制目录,但是,文件与目录的权限可能会被改变  也可以利用『 cp -a /etc /tmp 』来下达指令,尤其是在备份的情况下!  basename /etc/sysconfig/network <== 很简单!就取得最后的档名~  dirname /etc/sysconfig/network <== 取得的变成目录名了!  cat -n /etc/issue [ -n ] 显示行号 [ -b ] 显示行号，除去空行  [ -A ]显示特殊字符  tac /etc/issue 反向显示  nl -b a -n rz -w 3 /etc/issue nl (添加行号打印)  touch attrtest <==建立一个空文件  chattr +aiS attrtest [ S ] sync 任何修改『同步』写入磁盘中  [ a ] :只能增加数据,而不能删除修改  [ i ] 『不能被删除、改名、设定连结也无法写入或新增数据!』  lsattr attrtest  =====================================================================================which [-a] :由 PATH 目录中可以找到的指令均列出,而不止第一个被找到的指令名称  whereis [ -l ] :可以列出 whereis 会去查询的目录  [ -b ] :只找 binary 格式的文件  [ -m ] :只找在说明文件 manual 路径下的文件  [ -s ] :只找 source 来源文件  [ -u ] :搜寻不在上述三个项目当中的其他特殊文件  locate [-ir] keyword [ -i ] :忽略大小写的差异;  [ -c ] :不输出档名,仅计算找到的文件数量  [ -l ] :仅输出几行的意思,例如输出五行则是 -l 5  [ -S] :输出 locate 所使用的数据库文件的相关信息,包括该数据库纪录的文件/目录数量等  [ -r ] :后面可接正规表示法的显示方式  删除行末多余的空格  :%s/\s\+$//  折叠    **手动折叠**  zfap (ap 是一个文本对象, 可以将 zf 操作符跟任何一个移动命令联用)  zf F-old creation (创建折叠) zi 禁止，恢复折叠  zo O-pen a fold (打开折叠) zO 打开所有光标行上的折叠用  zc C-lose a fold (关闭折叠) zC 关闭所有光标行上的折叠用  zr 减少 (R-educe) 折叠 (嵌套折叠) zR 减少折叠直至一个也不剩  zm 折叠更多 (M-ore) zM 增加折叠，直至所有的折叠都关闭  zd 删除一个光标行上的折叠用 zD 删除所有光标行上的折叠用  对折叠的操作  :set foldcolumn=4 显示折叠栏  :set foldmethos=expr 依表达式折叠  :set foldmethod=marker 依标志折叠  :set foldmethod=indent 依缩进折叠  折叠未被改动的行  :setlocal diff foldmethod=diff scrollbind nowrap foldlevel=1 | |
|  | |
|  | |