



# Vim - the ubiquitous text editor

孟宁



关注孟宁

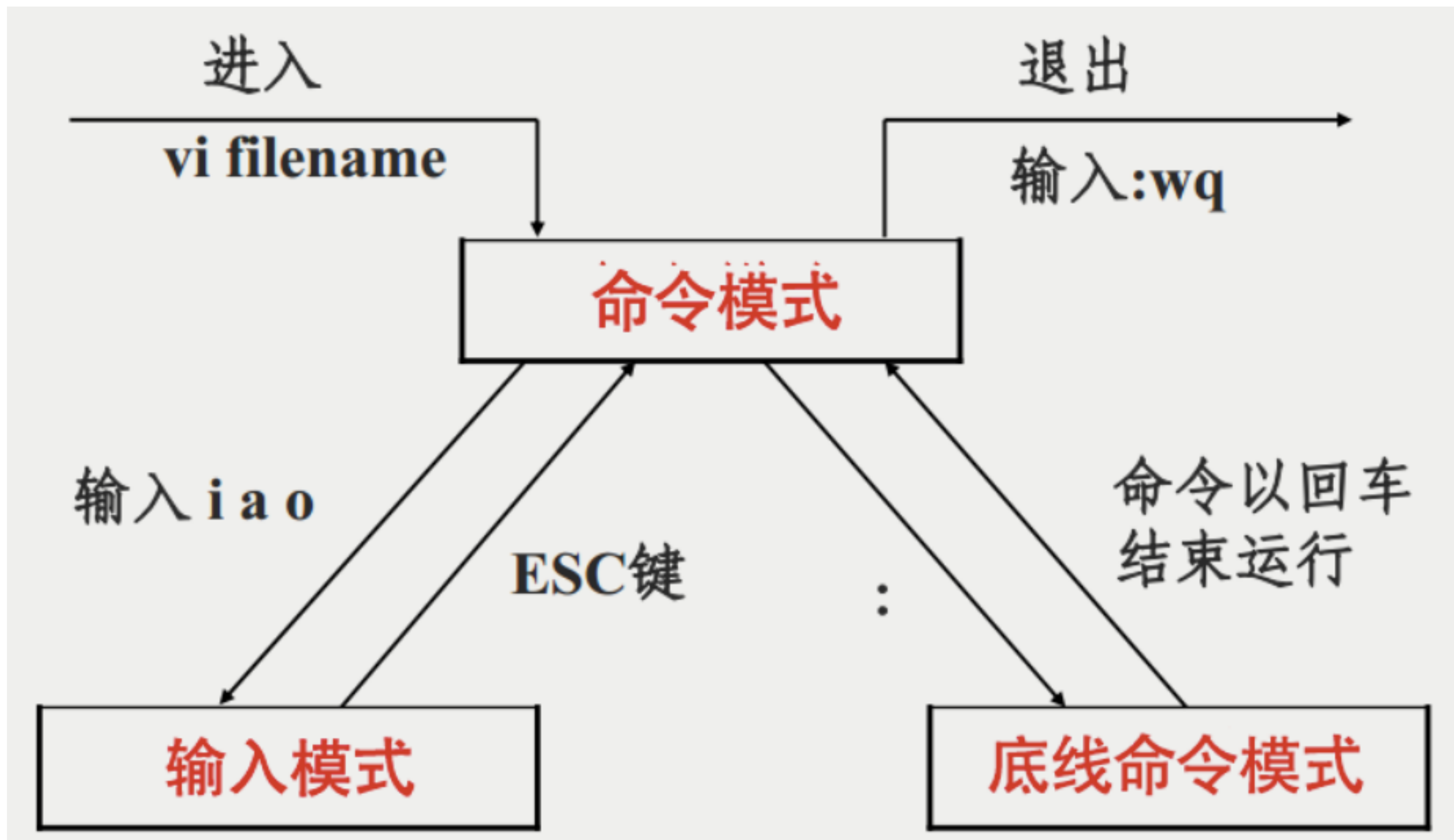
# Linux vi/vim

- 几乎所有的Unix-Like系统一般都会预装vi文本编辑器，其他的文本编辑器则不一定预装。
- vim具有程序编辑的能力，可以主动的以字体颜色辨别语法的正确性，方便程序设计。
- vim是从vi发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。
- 简单的来说，vi仅仅是文本编辑器，不过功能已经很齐全了。vim则是程序开发者的一项很好用的工具。连vim的官方网站 (<http://www.vim.org>) 也说vim是一个程序开发工具而不仅是文本编辑器。

# vi/vim的三种模式

- 命令模式 (Command mode) ，用户刚刚启动vi/vim，便进入了命令模式。此状态下敲击键盘动作会被vim识别为命令，而非输入字符。比如我们此时按下i，并不会输入一个字符，i被当作了一个命令。命令模式只有一些最基本的命令，因此仍要依靠底线命令模式输入更多命令
- 输入模式 (Insert mode) ，在命令模式下按下i就进入了输入模式，按ESC退出输入模式，切换到命令模式。
- 底线命令模式 (Last line mode) ，在命令模式下按下: (英文冒号) 就进入了底线命令模式。底线命令模式可以输入单个或多个字符的命令，可用的命令非常多。基本的命令有q (退出程序)、w (保存文件) 等。按ESC键可随时退出底线命令模式。

# vi/vim的三种模式



# 移动光标的方法

- h 或 向左箭头键(←) 光标向左移动一个字符
- j 或 向下箭头键(↓) 光标向下移动一个字符
- k 或 向上箭头键(↑) 光标向上移动一个字符
- l 或 向右箭头键(→) 光标向右移动一个字符
- + 光标移动到非空格符的下一行
- - 光标移动到非空格符的上一行
- 如果你将右手放在键盘上的话，你会发现 hjkl 是排列在一起的，因此可以使用这四个按钮来移动光标。如果想要进行多次移动的话，例如向下移动 30 行，可以使用 "30j" 或 "30↓" 的组合按键，亦即加上想要进行的次数(数字)后，按下动作即可！

# 移动光标的方法

- `n<space>` 那个 `n` 表示『数字』，例如 `20`。按下数字后再按空格键，光标会向右移动这一行的 `n` 个字符。例如 `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』：移动到这一行的最前面字符处(常用)
- `$` 或功能键[End] 移动到这一行的最后面字符处(常用)
- `H` 光标移动到这个屏幕的最上方那一行的第一个字符
- `M` 光标移动到这个屏幕的中央那一行的第一个字符
- `L` 光标移动到这个屏幕的最下方那一行的第一个字符
- `G` 移动到这个档案的最后一行(常用)
- `nG` `n`为数字。移动到这个档案的第 `n` 行。例如 `20G` 则会移动到这个档案的第 `20` 行(可配合 `:set nu`)
- `gg` 移动到这个档案的第一行，相当于 `1G` 啊！(常用)
- `n<Enter>` `n` 为数字。光标向下移动 `n` 行(常用)

# 翻页

- [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

# 删除、复制与粘贴

- `x, X` 在一行字当中，`x` 为向后删除一个字符 (相当于 `[del]` 按键)，`X` 为向前删除一个字符(相当于 `[backspace]` 亦即是退格键) (常用)
- `nx` `n` 为数字，连续向后删除 `n` 个字符。举例来说，我要连续删除 10 个字符，『`10x`』。
- `dd` 删除光标所在的那一整行(常用)
- `ndd` `n` 为数字。删除光标所在的向下 `n` 行，例如 `20dd` 则是删除 20 行 (常用)
- `d1G` 删除光标所在到第一行的所有数据
- `dG` 删除光标所在到最后一行的所有数据
- `d$` 删除光标所在处，到该行的最后一个字符
- `d0` 那个是数字的0，删除光标所在处，到该行的最前面一个字符



# 删除、复制与粘贴

- `yy` 复制光标所在的那一行(常用)
- `nyn` `n` 为数字。复制光标所在的向下 `n` 行，例如 `20yy` 则是复制 20 行(常用)
- `y1G` 复制光标所在行到第一行的所有数据
- `yG` 复制光标所在行到最后一行的所有数据
- `y0` 复制光标所在的那个字符到该行行首的所有数据
- `y$` 复制光标所在的那个字符到该行行尾的所有数据

# 删除、复制与粘贴

- p, P p为将已复制的数据在光标下一行贴上，P则为贴在光标上一行！ 举例来说，我目前光标在第 20 行，且已经复制了 10 行数据。则按下 p 后，那 10 行数据会贴在原本的 20 行之后，亦即由 21 行开始贴。但如果是按下 P 呢？ 那么原本的第 20 行会被推到变成 30 行。(常用)
- J 将光标所在行与下一行的数据结合成同一行
- c 重复删除多个数据，例如向下删除 10 行，[ 10cj ]
- u 复原前一个动作。(常用)
- [Ctrl]+r 重做上一个动作。(常用)
- 这个 u 与 [Ctrl]+r 是很常用的指令！ 一个是复原，另一个则是重做一次

# 搜索替换

- /word 向光标之下寻找一个名称为 word 的字符串。例如要在档案内搜寻 vbird 这个字符串，就输入 /vbird 即可！（常用）
- ?word 向光标之上寻找一个字符串名称为 word 的字符串。
- n 这个 n 是英文按键。代表重复前一个搜寻的动作。举例来说，如果刚刚我们执行 /vbird 去向下搜寻 vbird 这个字符串，则按下 n 后，会向下继续搜寻下一个名称为 vbird 的字符串。如果是执行 ?vbird 的话，那么按下 n 则会向上继续搜寻名称为 vbird 的字符串！
- N 这个 N 是英文按键。与 n 刚好相反，为『反向』进行前一个搜寻动作。例如 /vbird 后，按下 N 则表示『向上』搜寻 vbird 。
- 使用 /word 配合 n 及 N 是非常有帮助的！可以让你重复的找到一些你搜寻的关键词！

# 搜索替换

- `:n1,n2s/word1/word2/g` `n1` 与 `n2` 为数字。在第 `n1` 与 `n2` 行之间寻找 `word1` 这个字符串，并将该字符串取代为 `word2`！举例来说，在 100 到 200 行之间搜寻 `vbird` 并取代为 `VBIRD` 则：
- `『:100,200s/vbird/VBIRD/g』`。(常用)
- `:1,$s/word1/word2/g` 或 `:%s/word1/word2/g` 从第一行到最后一行寻找 `word1` 字符串，并将该字符串取代为 `word2`！(常用)
- `:1,$s/word1/word2/gc` 或 `:%s/word1/word2/gc` 从第一行到最后一行寻找 `word1` 字符串，并将该字符串取代为 `word2`！且在取代前显示提示字符给用户确认 (`confirm`) 是否需要取代！(常用)

# 切换到编辑模式

- i, I 进入输入模式(Insert mode):
  - i 为『从目前光标所在处输入』， I 为『在目前所在行的第一个非空格符处开始输入』。(常用)
- a, A 进入输入模式(Insert mode):
  - a 为『从目前光标所在的下一个字符处开始输入』， A 为『从光标所在行的最后一个字符处开始输入』。(常用)
- o, O 进入输入模式(Insert mode):
  - 这是英文字母 o 的大小写。o 为『在目前光标所在的下一行处输入新的一行』； O 为在目前光标所在处的上一行输入新的一行！(常用)
- r, R 进入取代模式(Replace mode):
  - r 只会取代光标所在的那一个字符一次； R 会一直取代光标所在的文字，直到按下 ESC 为止；(常用)
- [Esc] 退出编辑模式，回到一般模式中(常用)
- 编辑模式在vi画面的左下角处会出现『--INSERT--』或『--REPLACE--』的字样

# 命令模式

- :w 将编辑的数据写入硬盘档案中(常用)
- :w! 若文件属性为『只读』时，强制写入该档案。不过，到底能不能写入，还是跟你对该档案的档案权限有关啊！
- :q 离开 vi (常用)
- :q! 若曾修改过档案，又不想储存，使用！为强制离开不储存档案。
- 注意一下啊，那个惊叹号 (!) 在 vi 当中，常常具有『强制』的意思～
- :wq 储存后离开，若为 :wq! 则为强制储存后离开 (常用)
- ZZ 这是大写的 Z 喔！若档案没有更动，则不储存离开，若档案已经被更动过，则储存后离开！

# 命令模式

- `:w [filename]` 将编辑的数据储存成另一个档案（类似另存新档）
- `:r [filename]` 在编辑的数据中，读入另一个档案的数据。亦即将『filename』这个档案内容加到游标所在行后面
- `:n1,n2 w [filename]` 将 n1 到 n2 的内容储存成 filename 这个档案。
- `:! command` 暂时离开 vi 到指令行模式下执行 command 的显示结果！例如『`:! ls /home`』即可在 vi 当中察看 /home 底下以 ls 输出的档案信息！

# vim环境的变更

- `:set nu` 显示行号，设定之后，会在每一行的前缀显示该行的行号
- `:set nonu` 与 `set nu` 相反，为取消行号！



# 代码中批量添加注释

- 批量注释: Ctrl + v 进入块选择模式, 然后移动光标选中你要注释的行, 再按大写的 I 进入行首插入模式输入注释符号如 // 或 #, 输入完毕之后, 按两下 ESC, Vim 会自动将你选中的所有行首都加上注释, 保存退出完成注释。
- 取消注释: Ctrl + v 进入块选择模式, 选中你要删除的行首的注释符号, 注意 // 要选中两个, 选好之后按 d 即可删除注释, ESC 保存退出。
- 批量注释: 使用下面命令在指定的行首添加注释。使用名命令格式: :起始行号,结束行号s/^/注释符/g (注意冒号), 如:10,20s#^#//g, :10,20s/^/#/g
- 取消注释: 使用名命令格式: :起始行号,结束行号s/^注释符//g (注意冒号), 如:10,20s#^//##g, :10,20s/#//g

# vi / vim 键盘图

**Esc**  
 命令模式

~ 转换大小写	! 外部过滤器	@ 运行宏	# prev ident	\$ 行尾	% 括号匹配	^ "软"行首	& 重复:s	* next ident	( 句首	) 下一句首	"soft" bol down	+ 后一行行首
\. 跳转到标注	1	2	3	4	5	6	7	8	9	0 "硬"行首	- 前一行行首	= 自动 <sup>3</sup> 格式化
Q 切换至ex模式	W 下一单词	E 词尾	R 替换模式	T back 'till	Y 拷贝行	U 撤消行内命令	I 到行首插入	O 分段(前)	P 粘贴(前)	{ 段首	}	段尾
q 录制宏	w 下一单词	e 词尾	r 替换字符	t 'till	y 拷贝 <sup>1,3</sup>	u 撤消命令	i 插入模式	o 分段(后)	p 粘贴 <sup>1</sup> (后)	[ 杂项	]	杂项
A 在行尾附加	S 删除行并插入	D 删除至行尾	F 行内字符反向查找	G 文尾/行号	H 屏幕顶行	J 合并两行	K 帮助	L 屏幕底行	:	ex命令	" 寄存器 <sup>1</sup> 标识	行首/列
a 附加	s 删除字符并插入	d 删除 <sup>1,3</sup>	f 行内字符查找	g 附加命令 <sup>6</sup>	h ←	j ↓	k ↑	l →	:	重复	' 跳转到标注的行首	\ 未用!
Z 退出 <sup>4</sup>	X 退格	C 修改至行末	V 可视行模式	B 前一单词	N 查找上一处	M 屏幕中间行	< 反缩进 <sup>3</sup>	> 缩进 <sup>3</sup>	?	向前搜索		
Z 附加命令 <sup>5</sup>	X 删除(字符)	c 修改 <sup>1,3</sup>	v 可视模式	b 前一单词	n 查找下一处	m 设置标注	, u/T/f/F 反向	. 重复命令	/	向后搜索		

- 动作** 移动光标, 或者定义操作的范围
- 命令** 直接执行的命令, **红色命令** 进入编辑模式
- 操作** 后面跟随表示操作范围的指令
- extra** 特殊功能, 需要额外的输入
- q.** 后跟字符参数

**w,e,b命令**  
 小写(b): `quux(foo, bar, baz);`  
 大写(B): `quux(foo, bar, baz);`

**主要ex命令:**  
 :w (保存), :q (退出), :q! (不保存退出)  
 :e f (打开文件 f),  
 :%s/x/y/g ('y' 全局替换 'x'),  
 :h (帮助 in vim), :new (新建文件 in vim),

**其它重要命令:**  
 CTRL-R: 重复 (vim),  
 CTRL-F/-B: 上翻/下翻,  
 CTRL-E/-Y: 上滚/下滚,  
 CTRL-V: 块可视模式 (vim only)

**可视模式:**  
 漫游后对选中的区域执行操作 (vim only)

- 备注:**
- (1) 在 拷贝/粘贴/删除 命令前使用 "x (x=a..z,\*) 使用命令的寄存器('剪贴板') (如: "ay\$ 拷贝剩余的行内容至寄存器 'a')
  - (2) 命令前添加数字 多遍重复操作 (e.g.: 2p, d2w, 5i, d4j)
  - (3) 重复本字符在光标所在行执行操作 (dd = 删除本行, >> = 行首缩进)
  - (4) ZZ 保存退出, ZQ 不保存退出
  - (5) zt: 移动光标所在行至屏幕顶端, zb: 底端, zz: 中间
  - (6) gg: 文首 (vim only), gf: 打开光标处的文件名 (vim only)



史上最全Vim快捷键键位图

# 练习作业

- 用vim新建一个test.c文件
- 将xxx文件内容插入到test.c文件中
- 将当前文件中xxx字符串全部替换为yyy字符串
- 将当前文件中10-20行的代码注释掉
- 将2-3行代码复制粘贴10次