

Linux C 编程指北

郑海永

October 8, 2019

目录

1 编辑器 vi/vim	1
2 编译器 gcc/g++	2
3 调试器 gdb	3

1. 编辑器 vi/vim

```
1 $ vim tmp.txt
2 $ man vim
```

- 插入模式编辑文件
- 命令模式操作文件

2. 编译器 gcc/g++

```
1 $ vim hello.c
2 $ file hello.c
3 $ cat hello.c
```

1. 预处理

version 1.1
April 1st, 06

vi / vim graphical cheat sheet

Esc
normal mode

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	(begin sentence) end sentence	"soft" bol down	+ next line
\ goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= autoformat

Q ex mode	W next WORD	E end WORD	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	} end parag.
q record macro	w next word	e end word	r replace char	t 'till	y yank	u undo	i insert mode	o open below	p paste after	[misc] misc

A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	. ex cmd line	" reg. spec	bol/ goto col
a append	s subst char	d delete	f find char	g extra cmds	h ←	j ↓	k ↑	l →	; repeat t/T/f/F	' goto mk. bol	\ not used!

Z quit	X back-space	C change to col	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un-indent	> indent	? find (rev.)
Z extra cmds	X delete char	c change	v visual mode	b prev word	n next (find)	m set mark	reverse t/T/f/F	. repeat cmd	/ find

motion moves the cursor, or defines the range for an operator

command direct action command, if red, it enters insert mode

operator requires a motion afterwards, operates between cursor & destination

extra special functions, requires extra input

q. commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: `quux(fool bar baz)`

WORDS: `quux(fool bar baz)`

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)

:e f (open file f),

:%s/x/y/g (replace 'x' by 'y' filewide),

:h (help in vim), :new (new file in vim),

Other important comands:

CTRL-R: redo (vim),

CTRL-F/-B: page up/down,

CTRL-E/-Y: scroll line up/down,

CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

- (1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay\$ to copy rest of line to reg 'a')
- (2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)
- (3) duplicate operator to act on current line (dd = delete line, >> = indent line)
- (4) ZZ to save & quit, ZQ to quit w/o saving
- (5) zt: scroll cursor to top, zb: bottom, zz: center
- (6) gg: top of file (vim only), gG: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Windows.

图 1: VIM 键盘图

```
1 $ gcc -E hello.c -o hello.i
2 $ file hello.i
3 $ cat hello.i
```

2. 编译

```
1 $ gcc -S hello.i -o hello.s
2 $ file hello.s
3 $ cat hello.s
```

3. 汇编

```
1 $ gcc -c hello.s -o hello.o
2 $ file hello.o
3 $ cat hello.o
```

4. 链接

```
1 $ gcc hello.o -o hello
2 $ file hello
3 $ cat hello
```

5. 运行

```
1 $ ls -l hello*
2 $ ./hello
```

3. 调试器 gdb

```
1 $ vim swapNum.c
```

```
1 #include <stdio.h>
2
3 void Swap(int* number1,int* number2)
4 {
5     int tmp>(*number1);
6     (*number1)=(*number2);
7     (*number2)=tmp;
8 }
9
10 int main()
11 {
12     int x=2;
13     int y=3;
14     printf("%d,%d\n",x,y);
15     Swap(&x,&y);
```

```
16     printf("%d,%d\n",x,y);
17     retrun 0;
18 }
```

```
1 $ gcc -o swapNum -g swapNum.c
2 $ gdb swapNum
```

```
1 (gdb) run
```

```
1 (gdb) break 13
```

```
1 (gdb) run
```

```
1 (gdb) print x
```

```
1 (gdb) n
```

```
2 (gdb) n
```

```
1 (gdb) break 6
```

```
2 (gdb) c
```

```
1 (gdb) n
```

```
2 (gdb) n
```

```
3 (gdb) n
```

```
1 (gdb) print x
2 (gdb) print y
```

```
1 (gdb) q
```