# Linux C 编程指北

郑海永

October 8, 2019

## 目录



## 1. 编辑器 vi/vim

```
$ vim tmp.txt
$ man vim
```

- 插入模式编辑文件
- 命令模式操作文件

## 2. 编译器 gcc/g++

https://zhuanlan.zhihu.com/p/518632019

```
$ vim hello.c
$ file hello.c
$ cat hello.c
```

用来探测给定文件的类型

1. 预处理

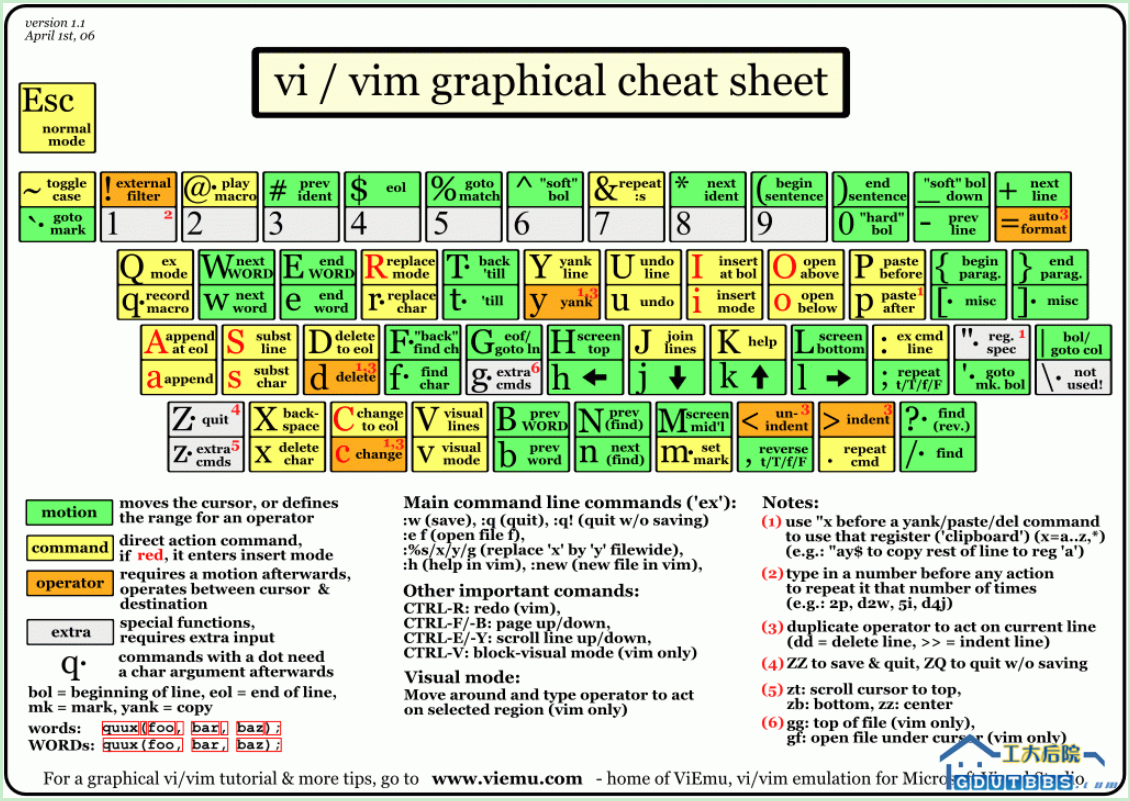
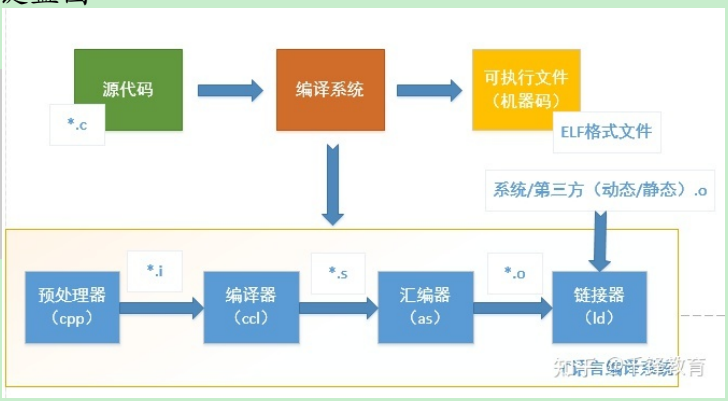



图 1: VIM 键盘图

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

源代码预处理，作用是将源程序文件中的预处理命令进行处理，如宏定义、文件包含等




2. 编译

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

编译过程就是对预处理完的文件进行一系列的词法分析，语法分析，语义分析及优化后生成相应的汇编代码

3. 汇编

```
$ gcc -c hello.s -o hello.o
$ file hello.o
$ cat hello.o
```

汇编过程生成处理器能识别的指令。每一个汇编语句几乎都对应一条处理器指令，通过调用Binutils中的汇编器as根据汇编指令和处理器指令的对照表一一翻译即可。

注意：目标文件已经是最终程序的某一部分了，但是在链接之前还不能执行。

可重定位目标文件（二进制），每个C代码文件可编译得到一个该类型文件，类似书桌组装原理

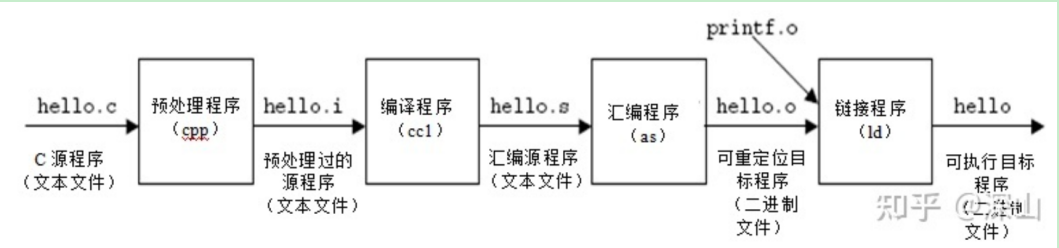
4. 链接

```
$ gcc hello.o -o hello
$ file hello
$ cat hello
```

将编译之后的所有可重定位目标文件，连同用到的静态库、运行时库，组合拼装成一个独立的可执行文件（二进制） --> 书桌

5. 运行

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




## 3. 调试器 gdb

```
$ vim swapNum.c
```

```c
#include <stdio.h>

void Swap(int* number1,int* number2)
{
    int tmp=(*number1);
    (*number1)=(*number2);
    (*number2)=tmp;
}

int main()
{
    int x=2;
    int y=3;
    printf("%d,%d\n",x,y);
    Swap(&x,&y);
```

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

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

如果想用调试器执行一个可执行文件， 在用gcc编译时必须加上-g选项

加载被调试的可执行程序文件。

```
1  (gdb) run
```

运行被调试的程序。
如果此前没有下过断点，则执行完整个程序；如果有断点，则程序暂停在第一个可用断点处。

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

设置断点。两可以使用“ 行号 ”“ 函数名称 ”“ 执行地址 ”等方式指定断点位置。

d: Delete breakpoint的简写，删除指定编号的某个断点，或删除所有断点。断点编号从1开始递增。

```
1  (gdb) run
```

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

显示指定变量（临时变量或全局变量）的值

```
1  (gdb) n
2  (gdb) n
```

执行一行源程序代码

```
1  (gdb) break 6
2  (gdb) c
```

Continue的简写，继续执行被调试程序，直至下一个断点或程序结束。

```
1  (gdb) n
2  (gdb) n
3  (gdb) n
```

```
(gdb) print x
(gdb) print y
```

```
(gdb) q
```

Quit的简写，退出GDB调试环境。