





```
ed f1          # 用编辑器创建 f1
a              # 添加 append
Ed is a line-oriented text editor of Unix.
It is old, simple and fun.
.              # 结束输入
w              # 存盘
q              # 退出
touch f{1..9}  # 更新 f1, 创建空文件 f2, ..., f9
```

```
mkdir d{1..9} # 创建多个目录
mkdir -p dir1/dir2/dir3 # 按路径创建多层目录
```



# 移动文件和目录

## 移动文件

```
mv f4 f4.1      # 将 f4 移动 (重命名) 为 f4.1
mv f5 d3         # 将文件移动到目录
mv -i f5 f6      # 覆盖时提示
mv f6 c/f6.1     # 移动并改名
mv *.1 d4        # 将文件移动到目录
```

## 移动目录

```
mv d4 dir1/dir2      # 目录移动
mv d5 d05             # 目录改名
```

## 删除文件和目录

## 删除文件

```
rm f8 f9      # 删除文件 f8、f9
rm -i f7      # 删除前确认
rm -f f*      # 强行删除
```

## 删除目录

```
rm -r d7 d8 # 删除空目录
rm -r -p dir1/dir2/dir3 # 删除多层空目录
rm -r d6 dir1 # 删除目录
rm -rf / # 删除所有文件!!!
```

# 判断文件类型

```
file
```

```
file /etc/hosts
```

```
file /usr/bin/cat
```

```
file /var
```

```
file /dev/sda1
```





- 功能强大，可根据各种复杂条件进行文件查找
- 对目录进行递归遍历，速度慢，应该尽量缩小查找范围

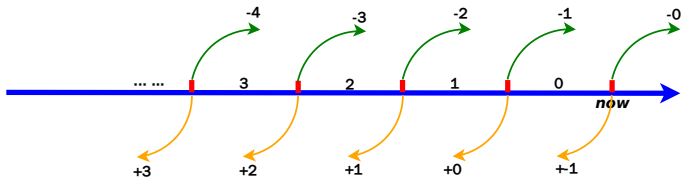
```
find /usr/bin -name chmod -print # 找文件 chmod
find /usr/bin -name "ch*" # 找 ch 开头的文件
find /usr/bin -size 5M # 找 5MB 大小的文件
find /usr/bin -size +5M # 找大于 5MB 的文件
find /usr/bin -size -5M # 找小于 5MB 的文件
find /usr/bin ! -size -5M # 找不小于 5MB 文件
find /usr/bin -size +5M -exec ls -sh {} \;
# 对找到的文件执行相应操作
find /usr/bin -size +5M -a -size -9M -exec ls -sh {} \;
# 多条件联合查找, -a(-and) 可省略, 还支持 -o(-or)、!(-not)
```

## 文件查找

```
find . -type -f -mtime +1 -exec ls -lt {} \;  
# 查找过去 48 小时之前修改过的文件  
find . -type -f -mtime 1 -exec ls -lt {} \;  
# 查找过去 24 小时之前, 48 小时之内修改过的文件  
find . -type -f -mtime -1 -exec ls -lt {} \;  
# 查找过去 24 小时之内修改过的文件  
find . -type -f -mtime +0 -exec ls -lt {} \;  
# 查找过去 24 小时之前修改过的文件  
find . -type -f -mtime 0 -exec ls -l {} \;  
# 查找现在之前, 过去 24 小时内修改过的文件  
find . -type -f -mtime -0 -exec ls -l {} \;  
# 查找现在之后修改过的文件  
find . -type -f -mtime +0 -mtime -2 -ok cp -l {} ~/bak \;  
# 查找过去 24 小时之前, 48 小时之内修改过的文件
```

# 查找文件

- 查找时间示意图 (图中每一小段代表 24 小时)



# 文件内容计数

## ● WC

### 示例

```
wc learn-vim
```

```
wc -l learn-vim
```

```
wc -w learn-vim
```

```
wc -c learn-vim
```

## 示例

```
grep root /etc/passwd
grep "insert mode" learn-vim
grep -n "insert mode" learn-vim
grep -vn "nologin" /etc/passwd
grep -c "/bin/bash" /etc/passwd
```

## 查看和设置登录 shell

### # 指定登录 *shell*, 下次登录生效

## 文件名通配符

\* 代表 0 到多个任意字符 (换行符除外)

```
ls file*
```

? 代表 1 个任意字符 (换行符除外)

```
ls file?
```

[xyz] 代表 1 个指定范围内字符

```
ls file[123]
```

```
ls file[0-9] [ab]
```

```
ls file[~0-9]
```

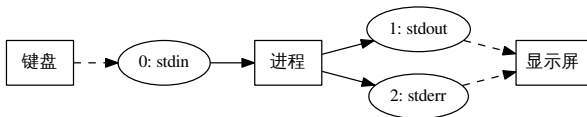
```
ls file[!23]
```

- 注意：文件名通配符只能匹配已经存在的文件名！

# 标准 I/O 文件

## ● shell 会为每个进程打开三个文件

- ① 标准输入 (stdin): 文件描述符为 0, 默认指向键盘
- ② 标准输出 (stdout): 文件描述符为 1, 默认指向显示屏
- ③ 标准错误 (stderr): 文件描述符为 2, 默认指向显示屏





# 标准输入重定向

## 示例

```
cat
```

```
cat <students.db
```

```
mail alice bob tom mary <letter
```

## 想一想

以下两条命令有何区别？

```
wc file
```

```
wc <file
```

# 标准输出重定向

## 示例

```
ls
ls >ls.out
cat file1 file2
cat file1 file2 >allfiles
cat file3 >>allfiles
who >temp
wc -l <temp
cat <file1 >file2
```

## 想一想

1. 为什么 `ls >ls.out` 会导致 `ls.out` 包括在名单中？
2. 执行 `wc temp >temp` 后，`temp` 文件的内容是什么？
3. 如果拼错了命令名，比如 `woh >temp`，会发生什么？

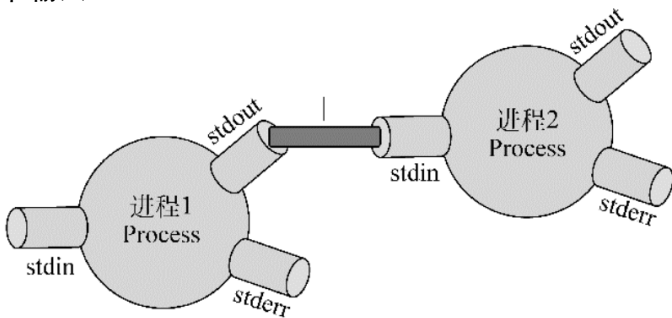
# 标准错误重定向

## 示例

```
ls .bashrc .profile >ls.out
ls .bashrc .profile 2>ls.err
ls .bashrc .profile 2>/dev/null # 抛弃错误输出
ls memo.1 memo.2 2>>ls.err
ls .bashrc .profile >ls.out 2>ls.err
ls .bashrc .profile >ls.out 2>&1
ls .bashrc .profile 2>&1 >ls.out
ls .bashrc .profile &>ls.out # 也可写成>&
ls .bash_profile memo.1 &>>ls.out
```

# 管道

- 通过管道可以将一个程序的标准输出连接到另一个程序的标准输入



## 示例

```
who | sort
```

# 管道

## 示例

```
who | wc -l
```

```
ls | wc -l
```

```
who | grep xiaobai | wc -l
```

## 想一想

下面两条命令有何不同？

```
who | sort
```

```
who >sort
```

## tee：T 形管道

```
who | tee who.out | grep xiaobai | tee grep.out | wc -l
```

```
1 mary female 19 100
2 tom male 20 93
3 susie female 20 97
4 bob male 21 90
5 alice female 18 92
6 mike male 19 89
```

```
cut -c3 students.db
cut -c3-7 students.db
cut -d' ' -f2,5 students.db
```

# sort：排序

## 示例文件：students2.db

```
All the light we cannot see^Anthony Doerr^2014^$15.29
Red queen^Victoria Aveyard^2015^$10.58
Saga,Vol.1^Brian K. Vaughan^2012^$8.46
The blood of Olympus^Rick Riordan^2014^$11.99
The day the crayons quit^Drew Daywalt^2013^$10.32
The girl on the train^Paula Hawkins^2015^$14.01
Winter(Lunar)^Marissa Meyer^2015^$13.84
```

## 示例

```
sort books.db # 按字典序排序
sort -t^ -k3 books.db # 按出版年份排序
sort -t^ -k4.2 books.db # 根据价格排序：-(
sort -t^ -k4.2 -nr books.db # 根据价格逆序排序：-)
```

# uniq：消除相邻重复行

## 示例文件：numbers

```
123
56
123
123
78
78
30
```

## 示例

```
uniq numbers
sort numbers | uniq
uniq -c numbers.sort
```



# join：行连接

## file1

```
1 tom
2 mary
3 bob
4 susie
5 alice
6 mike
```

## file2

```
tom 98
mary 100
alice 96
mike 90
susie 93
bob 88
```

## 示例

```
sort -k 2 file1 >file1.s
sort file2 >file2.s
join -12 -21 file1.s file2.s
```

# paste：列合并

## file1

```
1 tom
2 mary
3 bob
4 susie
5 alice
6 mike
```

## file3

```
male 23
female 22
male 21
female 20
female 19
male 20
```

## 示例

```
paste file1 file3
paste -d' ' file1 file3
```

# 压缩和解压

## gzip

```
gzip file1 file2          # 压缩
gzip -d file1.gz file2.gz # 解压缩
gunzip file1.gz file2.gz  # 解压缩
```

## bzip2

```
bzip2 file1 file2          # 压缩
bzip2 -d file1.bz2 file2.bz2 # 解压缩
bunzip2 file1.bz2 file2.bz2 # 解压缩
```

## zip

```
zip zipfile1 file1 file2 # 压缩文件 (不要加.zip 后缀)
zip -r zipdir1 dir1      # 压缩目录
unzip zipdir1.zip        # 解压缩
```

## tar

```
tar -cvf files.tar file1 file2      # 打包多个文件
tar -cvf dir1.tar dir1               # 打包目录
tar -tf files.tar                    # 查看包内容
tar -rf files.tar file3 file4        # 向包内添加文件
tar --delete -f files.tar file4      # 删除包内文件
tar -Af files.tar dir1.tar           # 合并包文件
tar -xvf boot.tar                    # 解包
tar -czvf boot.tar.gz /boot          # 打包并压缩 (gzip)
tar -xzvf boot.tar.gz                # 解压缩解包 (gzip)
tar -cjvf boot.tar.bz2 /boot         # 打包并压缩 (bzip2)
tar -xjvf boot.tar.bz2               # 解压缩解包 (bzip2)
```

# 分割文件

## split 用于将大文件分割为若干小文件

- b 文件大小 #定义文件分块的大小,单位为b(Byte)、k(KB)、m(MB)
- l 行数 #以行数为单位进行切割
- a n #指定文件名后缀长度为n
- d: #指定文件名后缀为数字

# 分割文件

## split

# 打包

```
tar -czvf linux2015.tar.gz linux2015fall/
```

# 分割

```
split -b 5m linux2015.tar.gz
```

```
split -b 5m -a 1 -d linux2015.tar.gz linux2015.tar.gz.part
```

# 合并还原

```
cat linux2015.tar.gz.part* >linux2015bak.tar.gz
```

## 图内藏文

```
cat desert.jpg secret >desert2.jpg
```

```
tail desert2.jpg
```

## cmp 比较两个文件是否相同

```
cmp file1 file2
```

## diff 比较两个文本文件

```
diff file1 file2    # 比较文件
```

```
diff -r dir1 dir2    # 比较目录
```

## comm 比较两个已排序文件

```
comm file1 file2
```

# 第一列输出仅出现在 *file1* 中的行

# 第二列输出仅出现在 *file2* 中的行

### # 第三列输出同时出现在 *file1* 和 *file2* 中的行

选项：-1/2/3 禁止输出第 1/2/3 列

# 散列密码

## md5sum

```
md5sum file1 file2 >filesun.md5 # 生成散列值
md5sum -c filesun.md5           # 检查完整性
```

## sha1sum

```
sha1sum file1 file2 >filesun.sha1 # 生成散列值
sha1sum -c filesun.sha1           # 检查完整性
```

## openssl passwd

```
# 生成用户加密口令
openssl passwd -1 -salt salt_string passwd
```



# tr：字符转换

- tr 只能通过 stdin 获取输入

## 示例

```
echo "WHAT are you DOING?" | tr 'A-Z' 'a-z'
echo "ButterFly" | tr 'a-zA-Z' 'n-za-mN-ZA-M'
tr '\t' ' ' <file.txt
echo "1a22bb333ccc" | tr -d '0-9'
echo "1a22bb333ccc" | tr -c -d '0-9\n'
echo "GNU is not UNIX." | tr -s ' '
tr -s '\n' <file.txt
```



```
vim          #启动vim
i           #进入insert模式
vim is a mode text editor. #输入文本
<ESC>       #回到normal模式
:w hello-vim #保存到指定文件
:q          #退出
```

# 入门操作

## 开始使用 vim(方式 2)

```
vim          #启动vim并进入normal模式
:e hello-vim #编辑指定文件
i           #进入insert模式
hello, vim!<ENTER> #输入文本
<ESC>       #回到normal模式
:wq         #进入command-line模式存盘并退出vim
:q!        #进入command-line模式不存盘并退出vim
```

## 开始使用 vim(方式 3)

```
vim hello-vim
```

## #启动vim并打开指定文件

G

## #光标移动到最后一行

O

## #在下面打开一个新行

vim has three basic modes.

## #输入文本

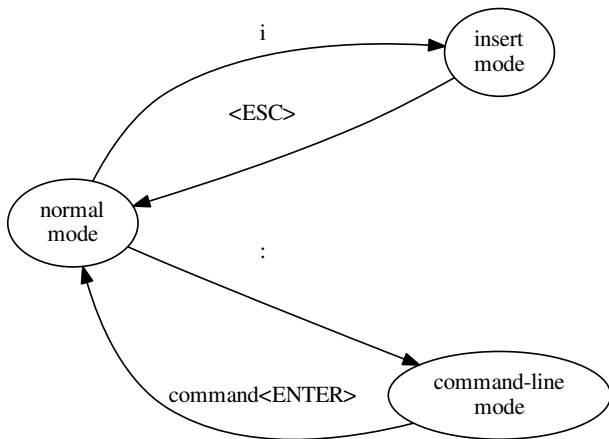
&lt;ESC&gt;

## #返回normal模式

ZZ

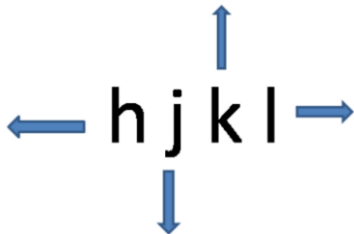
## #直接保存退出

# 入门操作



# 移动光标 (normal 模式)

j	#光标下移一行
k	#光标上移一行
h	#光标左移一个字符
l	#光标右移一个字符
5k	#光标下移5行
10h	#光标左移10个字符
^	#行首(文本)
0	#行首
\$	#行尾



vim 移动

# 移动光标

Ctrl-b        # 上一页 (backward)

Ctrl-f        # 下一页 (forward)

15G           # 光标移至第15行

gg            # 首行

G            # 尾行

H            # 屏幕首行 (high)

M            # 屏幕中间 (middle)

L            # 屏幕底行 (low)

Ctrl-o        # 回到上一位置

Ctrl-i        # 回到下一位置

Ctrl-g        # 查看当前光标位置



# 移动光标

b	#往前至单词开头 b (beginning)
w	#下一个单词 w (word)
e	#往后至单词尾部 e (end)
(	#上一句
)	#下一句
3(	#下3句
{	#上一段
}	#下一段
%	#配对括号

# 编辑命令

- i     #在光标前插入
- I     #在行首插入
- a     #在光标后插入
- A     #在行尾插入
- o     #在光标下方插入新行
- O     #在光标上方插入新行
- r     #替换光标所在字符(单字符替换, 且保持在normal模式)
- R     #从光标所在位置开始向后替换
- s     #删除光标所在字符并进入插入模式(可用多字符替换单字符)
- S     #删除当前行并进入插入模式(替换整行)
- x     #删除光标所在字符(向后删除, 且保持在normal模式)
- X     #删除光标前一字符(向前删除, 且保持在normal模式)

# 撤销与重做

## 撤销

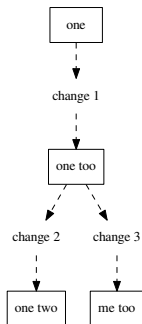
```
u          #撤销一次操作
U          #撤销对当前行的所有修改
:earlier 1f #回到上次存盘时的状态
```

## 重做 (要先有撤销操作才有效)

```
Ctrl+r      #重做最近被撤销一次操作
:redo 5     #重做最近撤销的5步操作
:later 1f   #进到下次存盘时的状态
```

# 撤销分支

## ● 示例



1. 在me too按u,回到one too

2. 再次按u,回到one

3. 按Ctrl-r,进到one two

4. 再按Ctrl-r,进到me too

u和Ctrl-r只能在当前分支内上下移动

g- #沿时间轴回退1步

g+ #沿时间轴前进1步

:undolist #查看undo分支树页节点列表

:undo 2 #回到分支2(one two)

:earlier 5m #回退至5分钟前的状态

:later 10s #前进至10秒后的状态

# 动词

## 动词代表操作

c #修改(change)  
d #删除(delete)  
r #替换(replace)  
y #复制(yank)  
v #选取(visual select)  
p #(paste),在光标右/下方粘贴  
P #(Paste),在光标左/上方粘贴  
> #缩进  
< #反缩进

## 行操作

cc #修改当前行  
dd #删除当前行  
yy #复制当前行  
3dd #删除3行  
5yy #复制5行  
4>> #缩进4行  
<< #反缩进当前行

# 名词和介词

## 名词代表文本对象

w	#一个单词(word)
s	#一个句子(sentence)
p	#一个段落(paragraph)
t	#一个标签(tag)
"	#一个"... "文本块
{	#一个{...}文本块

## 介词界定了范围或位置

i	#在...内(inside)
a	#环绕...(around)
t	#到...前(to)
f	#到...上(forward)

# 名词和介词

**iw: inside word**

we know **what** we are but know not what we may be.

**aw: around word**

we know **what** we are but know not what we may be.

**is: inside sentence**

we know what we are but know not what we may be. God be at your table.

**ip: inside paragraph**

we know what we are but know not what we may be. God be at your table.

# 组词为句

## 动词 + 介词 + 名词

```
dip #删除一个段落 : delete inside paragraph
vis #选取一个句子 : visual select inside sentence
ciw #修改一个单词 : change inside word
caw #修改一个单词 : change around word
dtx #删除文本直到字符x(不包括x) : delete to x
dfx #删除文本直到字符x(包括x) : delete forward x
>ap #缩进一个段落
>i{ #缩进一个{}块内的内容
```



# 数词：数词可修饰名词或动词

## 数词修饰名词：动词 + 数词 + 名词

c3w #修改3个单词：change three words

d2w #删除2个单词：delete two words

## 数词修饰动词：数词 + 动词 + 名词

2dw #两次删除单词：twice delete word

3x #三次删除字符：three times delete character

# 更多例子

## 动词 + 位置

dl #向右删除字符, 即x  
 dh #向左删除字符, 即X  
 yj #复制当前行, 即yy  
 yk #复制上一行  
 dfi #向右删除至字母i  
 2yj #同上  
 y2j #同上  
 dG #向下删除至结尾  
 c4G #修改至第4行  
 d\$ #删除至行尾  
 c^ #修改至行首

## 动词 + 名词

2cw #向右修改两个单词  
 d2b #向左删除两个单词  
 c) #修改至句尾  
 y{ #复制至段首  
 yaw #复制当前单词  
 2daw #删除两个单词  
 yas #复制当前句子  
 yap #复制光标所在段落  
 2yap #复制两个段落  
 da< #删除<...>  
 ci" #修改"... "内的...

# 可视模式

- 进入可视模式后可通过移动光标可用于选择大块文本

## 进入可视模式

v	#字符可视模式
Shift-v	#行可视模式
Ctrl-v	#块可视模式

- 对文本块的操作步骤：
  - 1 将光标移动至文本块的开始位置
  - 2 进入可视模式
  - 3 移动光标选择文本块
  - 4 按 c 修改, 按 y 复制, 按 d 剪切, 按 <ESC> 返回 normal 模式

# 其他编辑操作

- 转换大小写

~

- 将下一行与当前行合并

J

- 使用缩写

```
:ab nos network operating system
```

输入nos并按空格键

```
:iab cn computer networks
```

输入cn并按空格键

注意:iab缩写仅在insert模式中有效

# 搜索

## ● 前向搜索

/mode     #从当前位置向下搜索字符串mode  
?mode    #从当前位置向上搜索字符串mode  
n         #下一个  
N         #上一个

## ● 搜索光标所在单词

\*         #下一个  
#         #上一个

## ● 在当前行内搜索光标所在字符

fd        #下一个d  
Fd        #上一个d  
;         #重复f/F命令  
,         #反向重复f/F命令

# 搜索替换

<code>:s/old/new</code>	#将当前行第一次出现的old替换为new
<code>:s/old/new/g</code>	#将当前行所有的old替换为new
<code>:3,10s/old/new/g</code>	#将3到50行内所有old替换为new
<code>:%s/old/new/g</code>	#将当前文档所有的old替换为new
<code>:%s/old/new/gc</code>	#同上,但每次替换需要用户确认

# 重复上次编辑操作

## ● . 命令

- 从进入 insert 模式的命令开始到按 <ESC> 键回到 normal 模式中为止, 所有的动作 (其中没有发生过光标移动动作) 被当作一次操作, 可以用 . 重复执行
- 如果进入 insert 模式后做了多次光标移动动作, 则每两次光标移动之间的动作被当作一次操作, 因此按 <ESC> 键回到 normal 模式后, 用 . 只能重复 insert 模式中最后一次光标移动之后的所有动作
- 例: 在文件的每行开始增加" todo: "

```
gg      #光标移至首行
Itodo:  #在行首添加todo:
<ESC>  #回到normal模式
j.      #下一行重复上一步操作
j.      #同上
```

# 书签

- 有时需要临时离开当前编辑位置, 转到文档的其他位置进行编辑, 编辑完成后, 如何快速地从其他位置回到原来的位置呢?

## 设置书签

ma	#标记(mark)当前位置, 标记名可为a-zA-Z
G	#将光标移至末行
'a	#快速回到标记a所在行行首
`a	#快速回到标记a的准确位置



# 寄存器

- 复制或删除的内容可以分别放在多个寄存器中，将来可进行选择性粘贴

## 示例：使用寄存器实现选择性粘贴

```
"a4yy  #复制4行至寄存器a
"b2dd  #删除2行至寄存器b
"cyap  #复制1段至寄存器c
"ap     #粘贴寄存器a中的内容
"bp     #粘贴寄存器b中的内容
"cp     #粘贴寄存器c中的段落
```

# 录制与重放宏操作

- vim 可以把多步操作录制为单个宏，并可以播放宏以重复该多步操作

## 示例：为单词添加一对双引号

```
qa      #开始录制宏a
b       #移至单词开头
i"<ESC> #插入一个双引号
e       #移至单词结尾
a"<ESC> #追加一个双引号
q       #结束录制
        #将光标移至另一个单词
@a      #播放宏a
```

# 执行 shell 命令

## 在 vim 中执行 shell 命令

```
:!cal          #执行cal命令  
:sh            #暂时切换到shell,按Ctrl-d返回  
!!uname -r    #执行命令并用其输出替换当前行
```

# 文件处理

## 文件处理

```
:cd dir1          #切换到目录dir1
:pwd              #查看当前目录
:r file1          #在当前行后插入文件file1
:10 r file2       #在第10行后插入文件file2
:w file3          #将当前文件(另)存为file3
:4,20 w file4     #将4至20行另存为file4
```

# 文件加密

## 加密

```
:X (大写)  
:set key=123
```

## 解除加密

```
:set key=      #设置key值为空
```

# vim 与多文件处理

## 多文件

```
vim file1 file2 file3
:ls          #列出所有打开的文件
:b3         #跳至第3个文件
:n          #切换至下一文件
:N          #切换至上一文件
:e file2    #切换至指定文件
```

# vim 与多窗口处理

## 窗口操作

```
:new          #上下拆分
:vnew         #左右拆分
:sp/Ctrl-w s  #上下拆分
:vsp/Ctrl-w v #左右拆分
:close/:q     #关闭当前窗口
Ctrl-w j/k/h/l #上下左右跳转
Ctrl-w        #跳至下一窗口
Ctrl-w Ctrl-w #循环跳转
```

## 窗口操作

```
Ctrl-w o      #仅显示当前窗口
Ctrl-w =      #所有窗口等大
Ctrl-w _      #当前窗口最大化
Ctrl-w +      #增大窗口
Ctrl-w -      #减小窗口
:resize 3     #调整窗口行数
Ctrl-w r      #旋转窗口顺序
Ctrl-w K      #当前窗口置顶
```

# vim 与多标签处理

## 多标签

<code>:tabnew</code>	#创建新标签
<code>:tabe[dit] file</code>	#新标签编辑文件
<code>:tabc[lose]/:q</code>	#关闭标签
<code>gt</code>	#下一个标签
<code>gT</code>	#上一个标签
<code>:tabm[ove] n</code>	#当前标签移至位置n(从0开始)



# 获取帮助

## 获取帮助

`vimtutor`

`:h`

# vim 配置

## ● 临时配置

### :set 命令

```
:set nu[mber]      #显示行号
:set nonu[mber]    #取消显示行号
:set ai            #自动缩进(auto indent)
:set noai          #取消自动缩进
:set cindent shiftwidth=4 #设置C语言风格
:set all           #查看所有可用选项
```

## ● 永久配置

- 全局配置：/etc/vim/vimrc
- 用户配置：~/.vimrc

# .vimrc 配置文件示例

## C 编程配置

```
syntax on
set cindent
set nu
set tabstop=4
set shiftwidth=4
colo evening

map <F5> :call Run()<CR>
func! Run()
    exec "w"
    exec "!g++ -Wall % -o %<"
    exec "!./%<"
endfunc
```

