

02 图形界面与命令行

LINUX图形界面



图形界面

文本界面

```
Ubuntu 18.04.3 LTS lei-VirtualBox tty2
lei-VirtualBox login: lei
Password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-23-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

129 ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
75  ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦

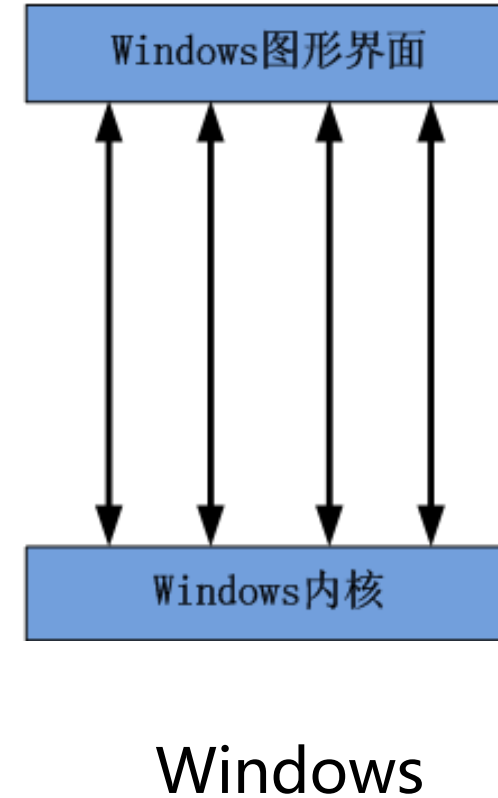
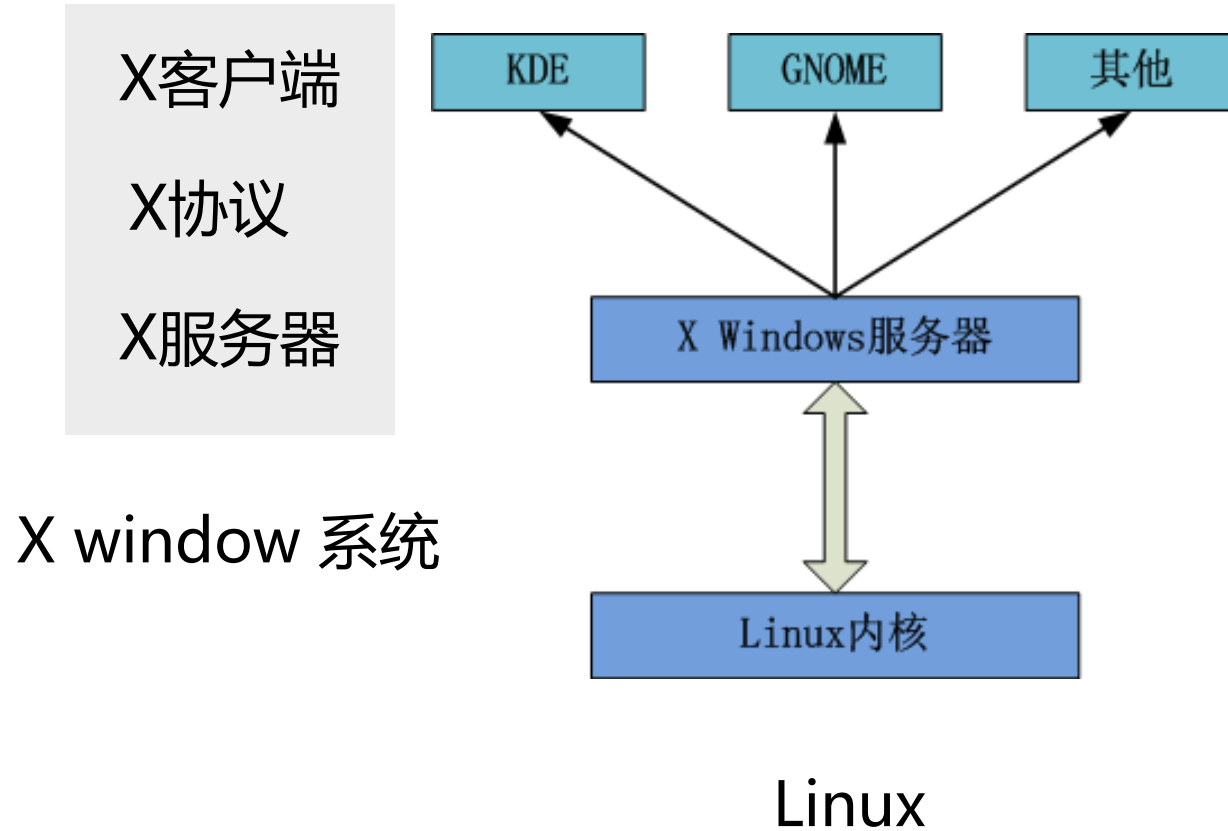
Your Hardware Enablement Stack (HWE) is supported until April 2023.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

lei@lei-VirtualBox:~$
```

LINUX图形界面

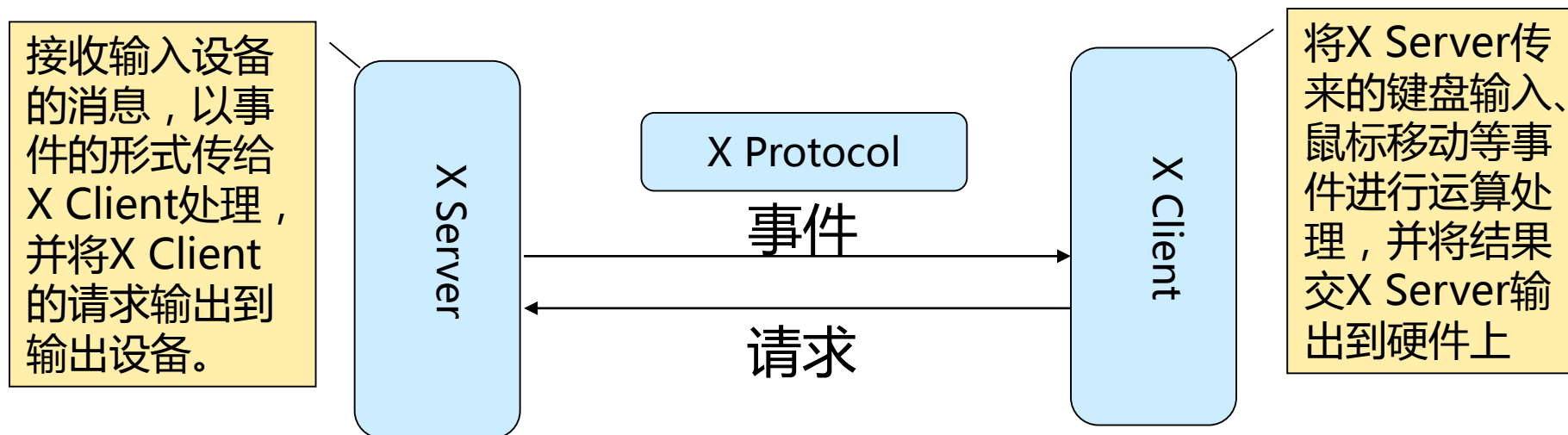


X WINDOW系统

- X Window系统是Linux的标准窗口环境。
- X Window是最底层的标准图形工具，它负责提供基本的图形操作，如打开窗口或显示图像。而窗口管理器负责窗口的有关操作。不同的窗口管理器在窗口显示方法上有所不同，但都是使用了X Windows图形工具。

X WINDOW系统核心概念

- X Window的核心概念是客户机/服务器（C/S）架构。
- X Server: 控制硬件，负责处理I/O（如创建窗口，在窗口中画图形，写文字等）
- X Client：与硬件无关，负责向server提出请求和进行程序运算
- X Protocol: X Server与X Client之间的通信协议。



C/S模式的好处

- X Client与硬件无关，当硬件更换时，无需改动X Client程序。方便了X Client在不同平台上移植。
- 远程操作：客户端可以在远程计算机上执行计算任务。
- 同一台计算机上，可以运行多个X Client。
- 单台计算机也可以运行多个X Server，供远程多个X Client操作。

X WINDOW系统核心概念

- X Window提供了不局限于一个系统的环境。应用程序可以在不同的服务器上或网络工作站上运行，并在网络其他部分的X windows终端或工作站上显示。即应用程序的运行和显示是分离的，这是windows 中没有的概念。
- X Window环境引入的另一个概念是窗口和界面的分离。在X Window环境下，必须运行两个应用程序才能提供完整的图形化用户界面。一个是X 服务器，用以建立图形显示、显示窗口、鼠标运动等，但不提供菜单、窗口边框和移动切换最大化、最小化窗口的机制，也没有菜单控件、漂亮的背景等，而这些由第二个应用程序即窗口管理器提供。

窗口管理器

- X window 系统只是提供了建立窗口的一个标准，具体的窗口形式由窗口管理器决定。
- 窗口管理器控制窗口的外观，并提供与用户交互的方法。
- 桌面环境包含窗口管理器。例如，FVWM，IceWM，Mutter，Kwin。

集成桌面环境

- X Window仅能负责桌面管理的工作（如窗口、菜单等）。
- 集成桌面环境除了可以启动X Windows，还可以充当应用程序的统一界面，包含了多种工具程序（如文字处理、多媒体、网络工具等）。
- 主流的Linux桌面环境：GNOME(GNU 网络对象模型环境)和KDE(K桌面环境)，Ubuntu默认使用的是Unity。
- Unity是基于GNOME桌面环境的用户界面，被设计得能更高效地使用屏幕空间，消耗的系统资源更少。

集成桌面环境

桌面环境包括会话程序，窗口管理器，面板和桌面程序。

会话程序：保证X图形组件正常运行，用于启动窗口管理器。

面板：提供用户交互，便于用户运行程序。

桌面程序：显示背景，窗口和桌面上的各种面板控件都在其上一层显示。

文本模式

在图形界面按组合键
<Ctrl> + <Alt> + <F(n)>
切换到文本控制台界面。

tty2-tty6对应五个虚拟控制台

注销
执行命令exit 或 logout

```
Ubuntu 18.04.3 LTS lei-VirtualBox tty2
lei-VirtualBox login: lei
Password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-23-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

129 ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
75  ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦

Your Hardware Enablement Stack (HWE) is supported until April 2023.

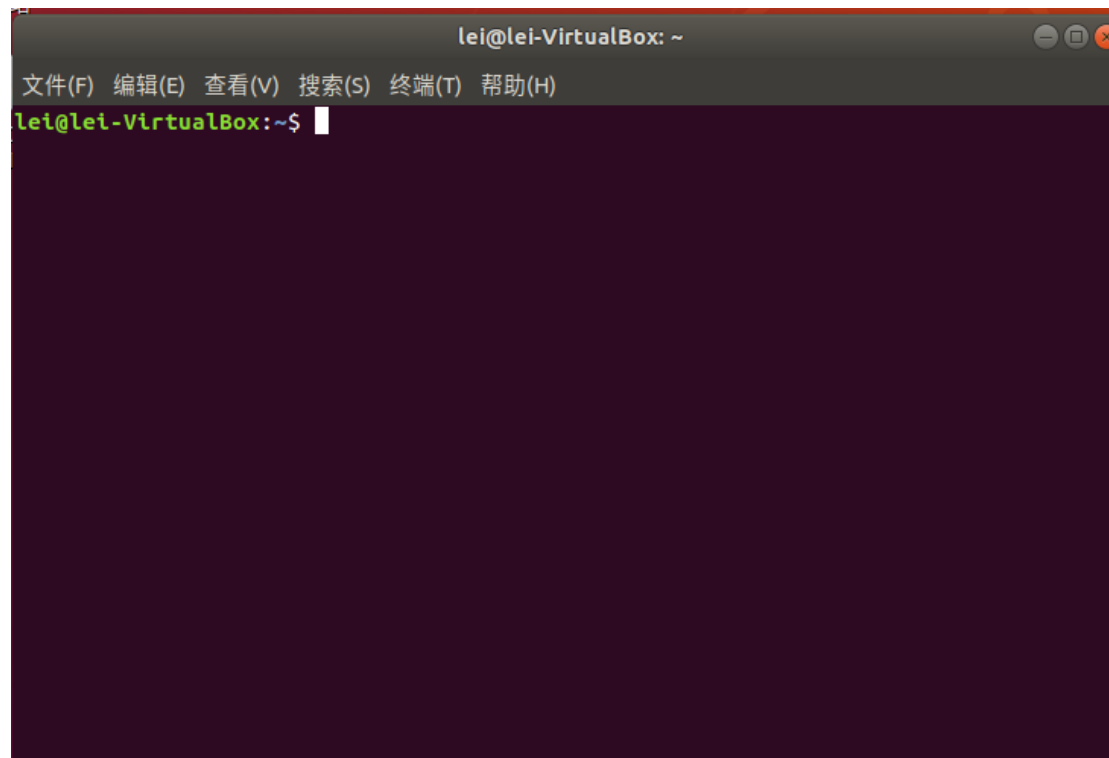
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

lei@lei-VirtualBox:~$
```

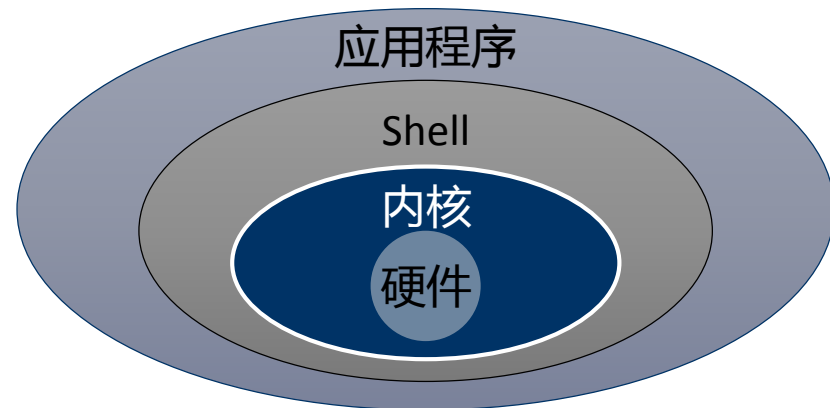
LINUX命令行界面

使用组合键<Ctrl> + <Alt> + T打开终端控制台



SHELL

- shell 是 Linux 系统中一个重要的层次，它是用户与系统交互的借口。
- shell 除了作为命令解释程序以外，还是一种高级程序设计(脚本)语言，它有变量、关键字，有各种控制语句，支持函数模块，有自己的语法结构。利用 shell 程序设计语言可以编写出功能很强但代码简单的程序。
- shell 的概念最初是在 UNIX 操作系统中形成和得到广泛应用的。UNIX 的 shell 有很多种类，Linux 系统集成了 UNIX 系统中 shell 的全部功能，现在默认使用的是bash。



SHELL的特点

- 对已有命令进行恰当组合，构成新的命令，而组合方式很简单。
- 提供了文件名扩展字符（通配符，如*，？，[]），使得用单一的字符串可以匹配多个文件名，省去键入一长串文件名的麻烦。
- 可以直接使用 shell 的内置命令，而不需创建新的进程，如 shell 中提供的 cd、echo、exit、pwd、kill 等命令。
- shell 允许灵活地使用数据流，提供通配符、输入/输出重定向、管道等机制，方便了模式匹配、I/O 处理和数据传输。

SHELL的特点

- 结构化的程序模块，提供了顺序流程控制、条件控制、循环控制等。
- shell 提供了在后台（&）执行命令的能力。
- shell 提供了可配置的环境，允许用户创建和修改命令、命令提示符和其他的系统行为。
- shell 提供了一个高级的命令语言，允许用户能创建从简单到复杂的程序。这些 shell 程序称为 shell 脚本。利用 shell 脚本，可把用户编写的可执行程序与 Linux 命令结合在一起，作为新的命令使用，从而便于用户开发新的命令。

SHELL的种类

- Linux 系统提供了多种不同的 shell。常用的有：
 - Bourne shell (简称 sh)
 - C-shell (简称csh)
 - Korn shell (简称 ksh)
 - Bourne Again shell (简称 bash)
- Linux 系统还包括其他一些流行的 shell , 如 ash、zsh 等。每种 shell 都有其特点和用途。

Bourne shell

- AT&T Bell 实验室的 Steven Bourne 为 AT&T 的 UNIX 开发的，它是 UNIX 的默认 shell，也是其他 shell 的开发基础。Bourne shell 在编程方面相当优秀，但在处理与用户的交互方面不如其他几种 shell。

BOURNE AGAIN SHELL

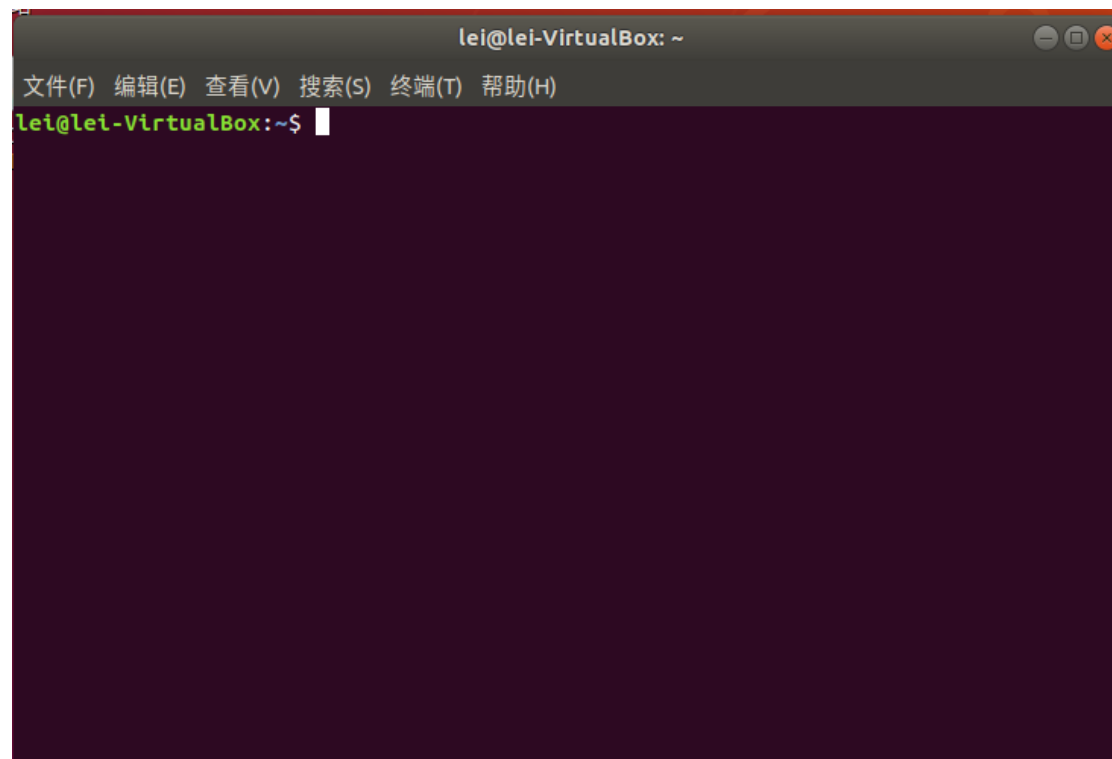
- bash 是自由软件基金会（GNU）开发的一个 shell，它是 Linux 系统中一个默认的 shell。bash 不但与 Bourne shell 兼容，还继承了 C shell、Korn shell 等下述优点。
 1. 命令行历史：使用命令行历史特性，可以恢复以前输入的命令。
 2. 命令行编辑：可以利用编辑器（如 vi）修改已经键入的命令。
 3. 命令补全：能在输入文件名的一部分之后，由系统自动填入剩余的部分。
 4. 别名扩展：能建立代表某些命令的名字。

使用SHELL

打开终端时就已经自动运行了一个默认的Shell程序。

在提示符后输入字符，Shell将对其进行解释。

```
echo $SHELL
```



环境变量

环境变量（ environment variables ）一般是指在操作系统中用来指定操作系统运行环境的一些参数。

PATH , HOME , LOGNAME , SHELL

echo \$HOME

LINUX命令语法

打开终端，可以看到SHELL提示符（管理员为#，普通用户为\$），在后面输入命令及选项和参数。

`command` [选项] [文件或目录列表]

ls -l

命令行基本用法

1. 编辑修改命令行

2. 调用历史命令

history

3. 自动补全

<Tab>

4. 一行多条命令和命令行续行

; 和 \

5. 强制终端命令运行

<Ctrl> + c

6. 获得帮助

man 命令名

SHELL中的特殊字符

" 单引号

括起来的字符视为普通字符串，包括空格、\$、/、\等特殊字符。

例如：echo '\$PATH'

输出 \$PATH

"" 双引号

括起来的字符串除\$、\、单引号、双引号仍作为特殊字符保留特征功能外，其它都视为普通字符。

例如：echo "\$PATH"

输出/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:

/usr/local/games:/snap/bin

SHELL中的特殊字符

"" 双引号

"" 中的\ 是转义符，Shell不会对其后的字符进行特殊处理，要将\$、\、单引号、双引号作为普通字符，在其前面加上\即可。

例如：echo "\"

```
lei@lei-VirtualBox:~$ echo "\"  
>
```

Shell等待继续输入。

` 反引号

括起来的字符被Shell解释为命令行，在执行时首先执行该命令行，并以运行结果代替该命令行。

例如：可以设变量a=`ls`

```
lei@lei-VirtualBox:~$ a=`ls`  
lei@lei-VirtualBox:~$ echo $a  
公共的 模板 视频 图片 文档 下载 音乐 桌面 examples.desktop hello2.py hello.py newfile.py  
lei@lei-VirtualBox:~$
```


输入输出重定向

重定向操作符

> 将命令输出写入到文件或设备（如打印机），而不是终端。

用法：命令 > 文件名

例如：ls > output

< 从文件而不是从键盘读入命令输入。

用法：命令 < 文件名

例如：wc < examplefile

>> 将命令输出添加到文件末尾而不删除文件中已有的信息。

用法：命令 >> 文件名

输入输出重定向

一般情况下，每个 Linux 命令运行时都会打开三个文件：

标准输入文件(stdin)：stdin的文件描述符为0，Linux程序默认从stdin读取数据。

标准输出文件(stdout)：stdout 的文件描述符为1，Linux程序默认向stdout输出数据。

标准错误文件(stderr)：stderr的文件描述符为2，Linux程序会向stderr流中写入错误信息。

如果希望 stderr 重定向到 file，可以这样写：

```
command 2 > file
```

如果希望将 stdout 和 stderr 合并后重定向到 file，可以这样写：

```
command > file 2>&1
```

管道

管道用于将一个命令的输出作为另一个命令的输入，使用符号 ‘|’ 连接命令。

例如显示当前目录下包含ab的文件和目录

```
ls | grep "ab"
```

查看bash的进程号

```
ps aux | grep bash
```

查看文件夹中文件数

```
ls | wc -l
```

命令替换

将一个命令的输出作为另一个命令的参数。

格式：命令1 `命令2`

例如：wc -l `ls`

执行多条命令

顺序执行

在执行时，以分号隔开的各条命令从左到右依次执行

- `pwd ; who | wc -l ; cd /usr/bin`

```
lei@lei-VirtualBox:/usr/bin$ pwd ; who | wc -l ; cd /usr/bin
/usr/bin
1
```

正则表达式

正则表达式 (re) 是一种可以用于模式匹配和替换的工具。

通配符

* 匹配 0 或多个字符

a*b a与b之间可以有任意长度的任意字符, 也可以一个也没有, 如aabcb, axyzb, a012b, ab。

例如显示当前文件夹下所有python文件 `ls *.py`

? 匹配任意一个字符

a?b a与b之间必须也只能有一个字符, 可以是任意字符, 如aab, abb, acb, a0b。

正则表达式

[list] 匹配 list 中的任意单一字符

a[xyz]b a与b之间必须也只能有一个字符, 但只能是 x 或 y 或 z,
如: axb, ayb, azb。

[!list] 匹配 除list 中的任意单一字符

a[!0-9]b a与b之间必须也只能有一个字符, 但不能是阿拉伯数字,
如axb, aab, a-b。

[c1-c2] 匹配 c1-c2 中的任意单一字符 如 : [0-9] [a-z]

a[0-9]b 0与9之间必须也只能有一个字符 如a0b, a1b... a9b。

VIM安装

在终端中运行： `sudo apt install vim`

```
lei@lei-VirtualBox:~$ sudo apt install vim
[sudo] lei 的密码:
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列软件包是自动安装的并且现在不需要了:
  linux-headers-5.0.0-23 linux-headers-5.0.0-23-generic linux-image-5.0.0-23-generic
  linux-modules-5.0.0-23-generic linux-modules-extra-5.0.0-23-generic
使用'sudo apt autoremove'来卸载它(它们)。
将会同时安装下列软件:
  vim-runtime
建议安装:
  ctags vim-doc vim-scripts
下列【新】软件包将被安装:
  vim vim-runtime
升级了 0 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，有 52 个软件包未被升级。
需要下载 6,789 kB 的归档。
解压缩后会消耗 33.0 MB 的额外空间。
您希望继续执行吗？ [Y/n] Y
```

输入Y，开始下载安装。

VIM安装

如果出现以下错误，无法获得锁，很可能是在未完成下载的情况下将窗口关闭，apt-get进程并没有结束而导致的。

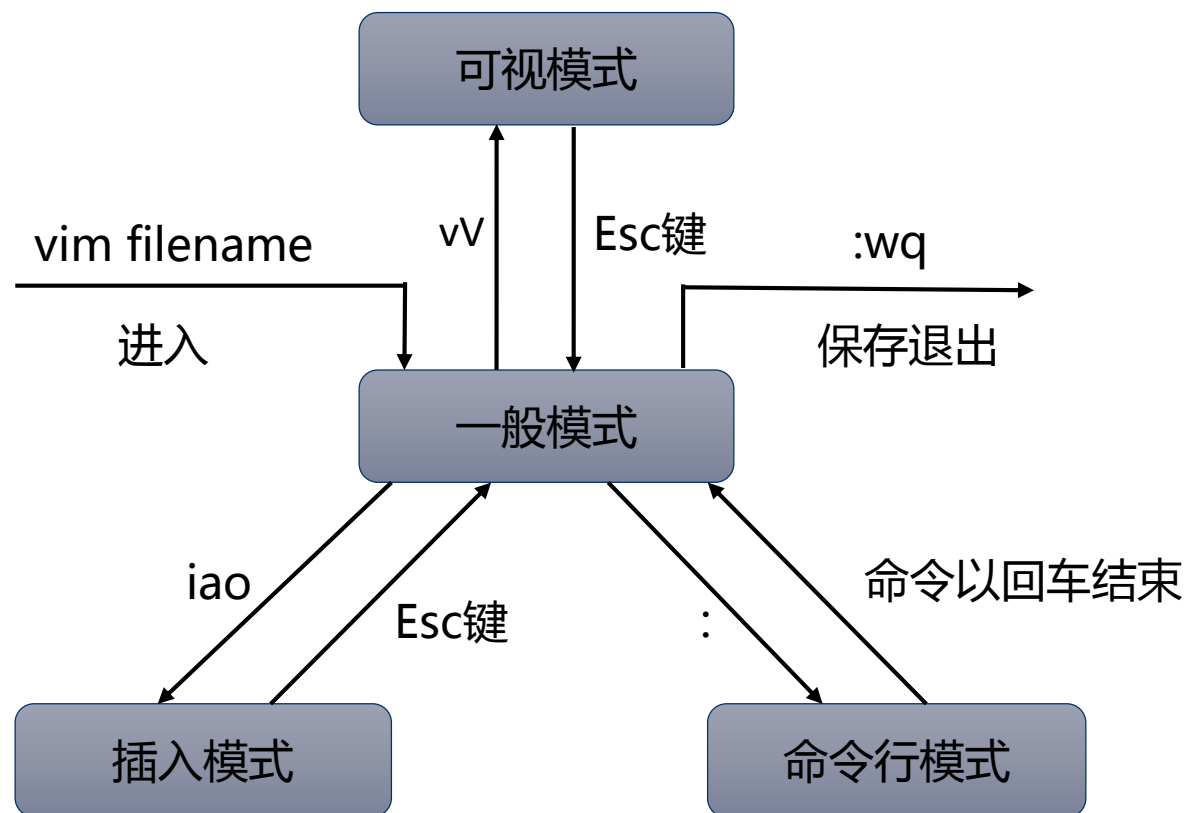
```
lei@lei-VirtualBox:~$ sudo apt-get install vim
[sudo] lei 的密码:
E: 无法获得锁 /var/lib/dpkg/lock-frontent - open (11: 资源暂时不可用)
E: 无法获取 dpkg 前端锁 (/var/lib/dpkg/lock-frontent)，是否有其他进程正占用它?
lei@lei-VirtualBox:~$
```

解决方法：

```
sudo rm /var/cache/apt/archives/lock
sudo rm /var/lib/dpkg/lock
```

重启Ubuntu，在终端中再次执行：`sudo apt-get install vim`

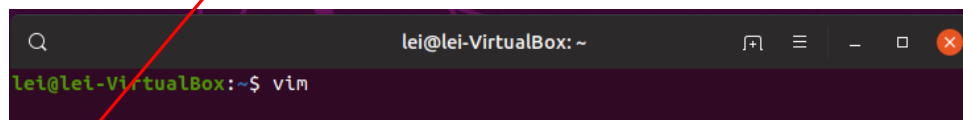
VIM四种模式



启动VIM

- 在终端中执行命令vim，即可进入vim一般模式。
- 在终端中执行命令vim filename，如果文件filename存在则打开filename，进入vim一般模式。

行首的~表明此行为空行。



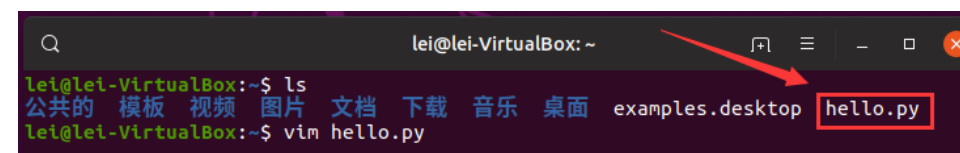
```
lei@lei-VirtualBox: ~$ vim
```



```
VIM - Vi IMproved
      版本 8.1.320
    维护人 Bram Moolenaar 等
  修改者 team+vim@tracker.debian.org
    vim 是可自由分发的开放源代码软件

    赞助 vim 的开发!
  输入 :help sponsor<Enter>  查看说明
  输入 :q<Enter>              退出
  输入 :help<Enter> 或 <F1>   查看在线帮助
  输入 :help version8<Enter> 查看版本信息

0,0-1 全部
```



```
lei@lei-VirtualBox: ~$ ls
公共的 模板 视频 图片 文档 下载 音乐 桌面 examples.desktop hello.py
lei@lei-VirtualBox: ~$ vim hello.py
```



```
print("hello world")

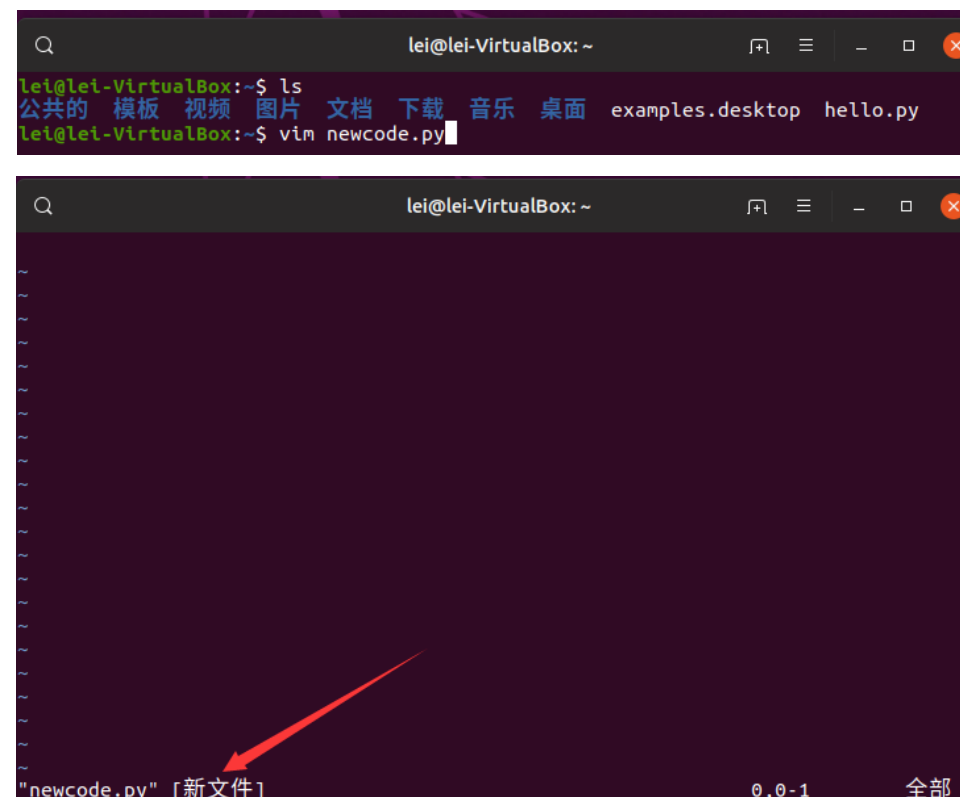
"hello.py" 1L, 21C 1,1 全部
```

启动VIM

- 如果执行vim filename, 文件filename不存在，则创建filename，进入vim一般模式。

例如，newcode.py文件不存在，执行
vim newcode.py则创建并打开该文件，进入
vim 一般模式。

注意：一般模式下不能输入内容，**输入内容**需要
切换到**插入**模式。



插入模式

在一般模式下，输入i, a, o, I, A, O可以进入插入模式，即可像在记事本中一样编辑内容。
按 ESC 回到一般模式。

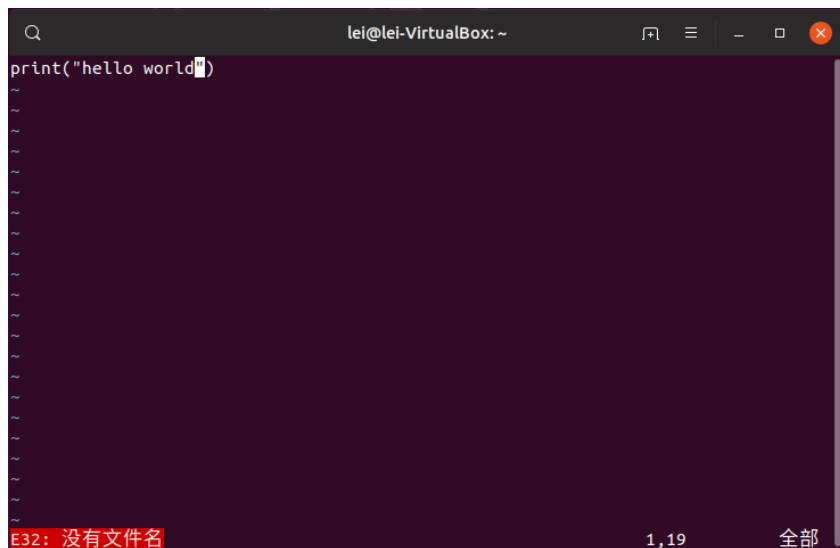
命令	说明
i	在当前光标处进行编辑
a	在光标后插入编辑
o	在当前行后插入一个新行
I	在行首插入
A	在行末插入
O	在当前行前插入一个新行

自动补齐： <C-n> 和 <C-p>

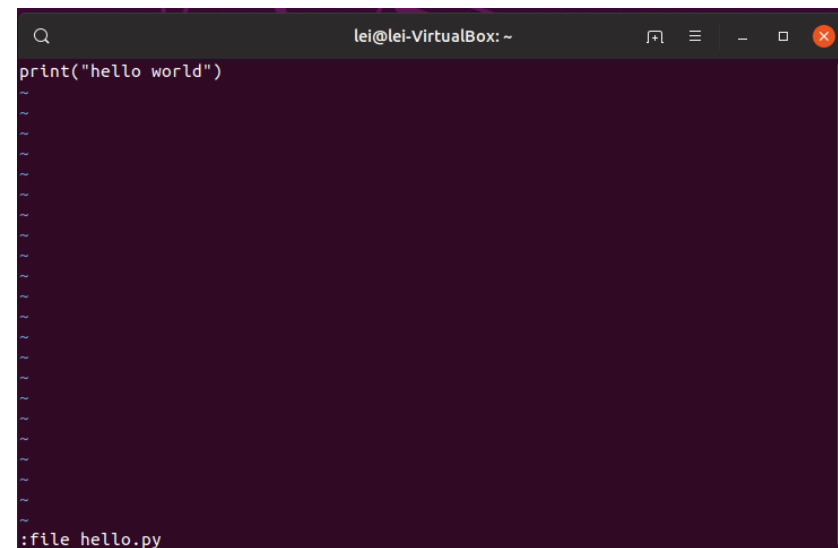
在插入模式下，可以输入一个词的开头，然后按 <C-p>或是<C-n>，就可以使用自动补齐功能。

保存未命名文件

如果直接打开vim进行编辑，末行模式输入:wq保存文件退出时会显示没有文件名，我们可以输入:file filename指定文件名，然后输入:wq保存退出。
也可以输入:w filename，进行保存，然后输入:q退出。



A terminal window titled 'lei@lei-VirtualBox: ~' showing a vim editor. The editor contains the text 'print("hello world")'. The status bar at the bottom left displays 'E32: 没有文件名' (E32: No filename), indicating that the file has not been named yet. The status bar also shows '1,19' and '全部'.



A terminal window titled 'lei@lei-VirtualBox: ~' showing a vim editor. The editor contains the text 'print("hello world")'. The status bar at the bottom left displays ':file hello.py', indicating that the file has been named 'hello.py'. The status bar also shows '1,19' and '全部'.

一般模式下编辑文本

按 ESC 回到一般模式

复制当前行 yy

粘贴 p和P都可以，p是表示在当前位置之后，P表示在当前位置之前

删除 dd 删除当前行，并把删除的行存到剪贴板里

dt+**字符**→ 删除当前行**字符**前的所有的内容

删除单个字符 x

Undo u

Redo <Ctrl>+r

命令行模式

一般模式下输入: 进入命令行模式, 然后输入命令回车即可执行命令。

另存为 :saveas <path/to/file>

存盘 :w

保存并退出 :wq

退出不保存 :q!

强行退出所有的正在编辑的文件 :qa!

打开一个文件 :e <path/to/file>

在打开的多个文件间切换 :bn 和 :bp

帮助 :help <command> 也可以就输入 :help 而不跟命令。退出帮助 :q

光标移动

一般模式下，k 上移；j 下移；h 左移；l 右移。

一般模式和插入模式，都可使用光标键 (←↑→)。

按键	说明
h	左
l	右 (小写L)
j	下
k	上

0：数字零，移动到行开头

^：到本行第一个不是空字符的位置（所谓空字符就是空格，tab，换行，回车等）

\$：到本行行尾

g_（字母g+下划线）：到本行最后一个不是blank字符的位置

NG：到第 N 行（注意命令中的G是大写的）

gg：到第一行。（相当于1G，或:1）

G：到最后一行。

光标移动

% : 匹配括号移动，包括 (, {, [. (需要把光标先移到括号上)

* 和 #: 匹配光标当前所在的单词，移动光标到下一个 (或上一个) 匹配单词
(*是下一个，#是上一个)

重复执行

. : (小数点) 可以重复上一次的命令

N<command> : command执行N次

例如 : 2dd : 删除2行

3p : 粘贴文本3次

查找替换

查找 /pattern : 搜索 pattern 的字符串 (如果搜索出多个匹配, 可按n键到下一个)

替换 :s (substitute) 命令用来查找和替换字符串。语法 :
:{作用范围}s/{目标}/{替换}/{替换标志}

例如 :%s/foo/bar/g会在全局范围(%)查找foo并替换为bar, 所有出现都会被替换 (g)

作用范围 : 作用范围分为当前行、全文、选区等等。

当前行 :s/foo/bar/g

全文 :%s/foo/bar/g

选区, 在Visual模式下选择区域后输入:, Vim即可自动补全为 : '<,>'。

: '<,>' s/foo/bar/g

5-11行 :5,11s/foo/bar/g

可视模式

在一般模式下按v，V或<Ctrl>+v进入可视模式，然后移动光标对文本进行选择。

v 进入字符可视化模式

V 进入行可视化模式

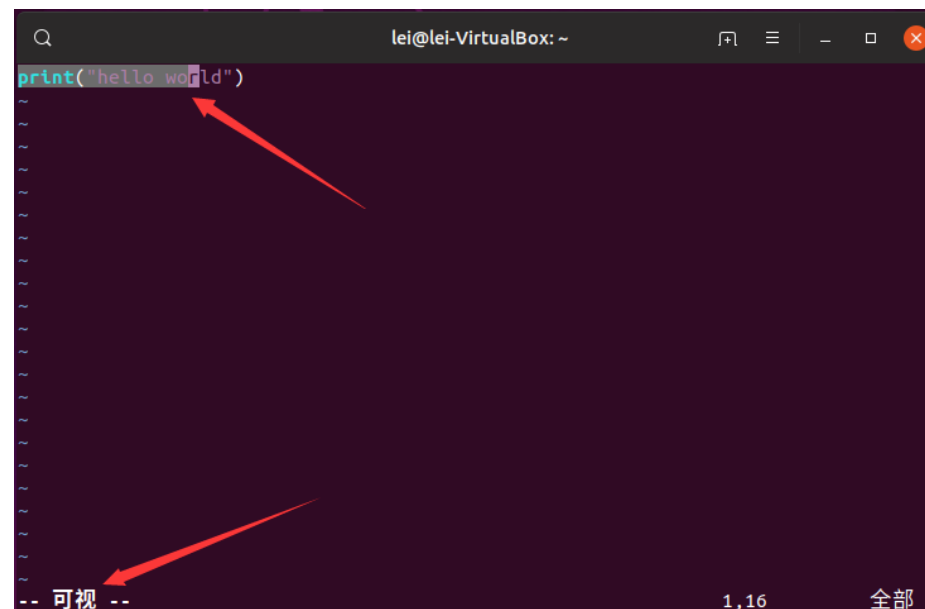
Ctrl+v 进入块可视化模式

选择好文本以后，可以进行复制粘贴操作，也可以执行

J：把所有的行连接起来（变成一行）

<shift>+< 或 >：左右缩进

=：自动缩进



可视模式示例

多行注释

^：移动到行开头

<Ctrl> + v：开始块操作

j：向下移动，选择要注释的行

I：进入在行开头插入模式

#：注释符

按ESC键来为每一行生效

在多行的行尾加内容

<Ctrl> + v：开始块操作

j：向下移动，选择要加注释的行

\$：移动到行末

A：进入在行末插入模式

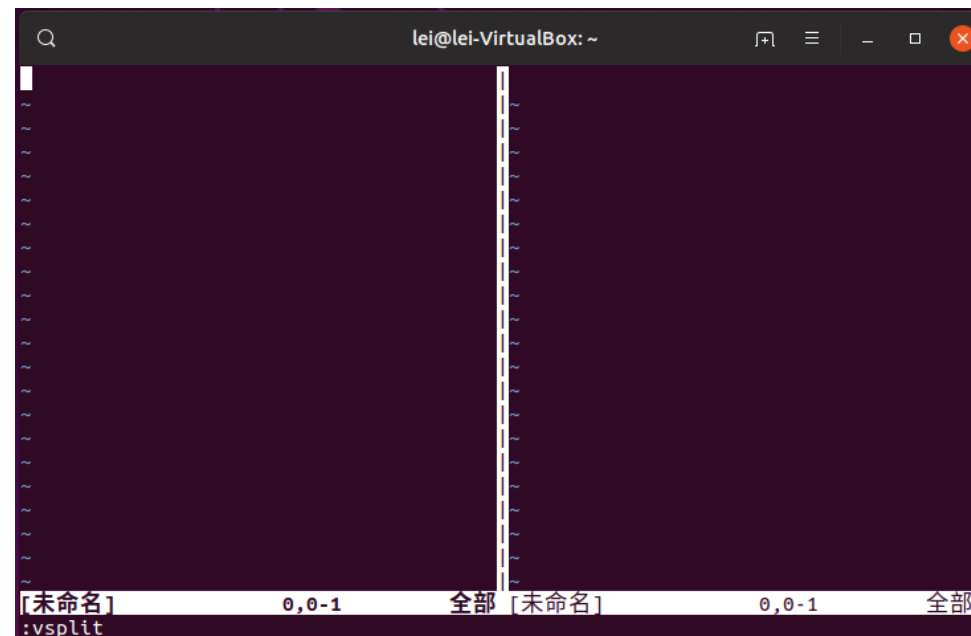
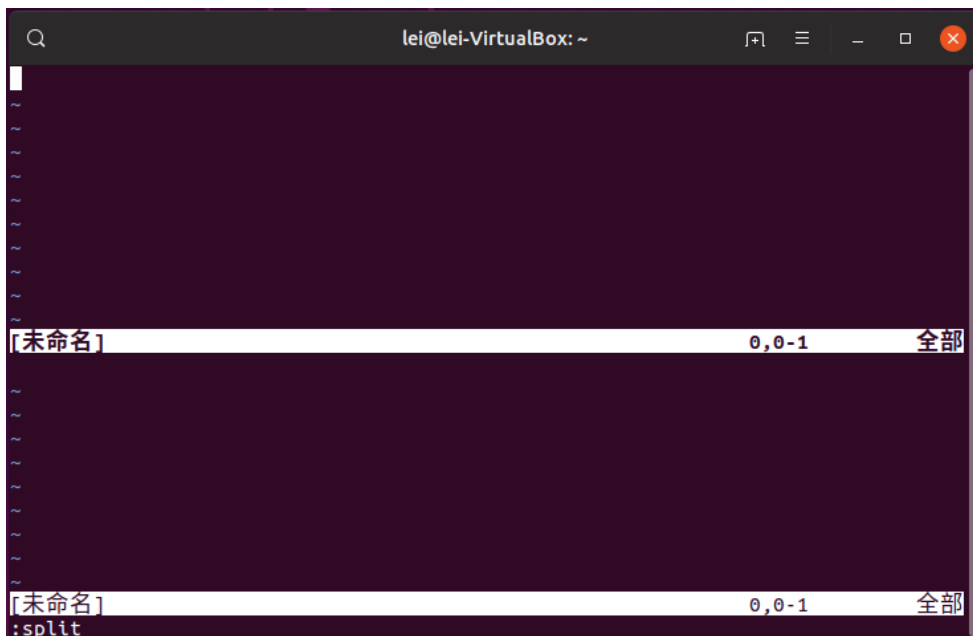
输入字符串

按 ESC键来为每一行生效

分屏显示

分屏: :split 和 :vsplit.

- :split → 创建水平分屏 (:vsplit创建垂直分屏)
- <Ctrl>+w+<dir> : dir就是方向, 可以是 hjkl 或是 ←↓↑→ 中的一个, 其用来切换分屏。
- <Ctrl>+w+_ : 水平分屏时最大化尺寸(或 <Ctrl>+w+| 垂直分屏时)
- <Ctrl>+w> ++ : 增加尺寸(或 <Ctrl>+w+- 减小尺寸)



VIM配置

Vim 的全局配置一般在/etc/vim/vimrc或者/etc/vimrc，对所有用户生效。

用户个人的配置在~/.vimrc。

如果只对单次编辑启用某个配置项，可以在命令行模式下，先输入一个冒号，再输入配置。举例来说，set number这个配置可以写在.vimrc里面，也可以在命令行模式输入。

`:set number`

配置项一般都有"打开"和"关闭"两个设置。"关闭"就是在"打开"前面加上前缀"no"。

" 打开 `set number` " 关闭 `set nonumber`

VIM配置

```
"自动缩进  
set autoindent  
set cindent  
set smartindent imap
```

```
"Tab键的宽度  
set tabstop=4
```

```
"统一缩进为4  
set softtabstop=4  
set shiftwidth=4
```

```
"搜索逐字符高亮  
set hlsearch  
set incsearch
```

总结

