# Linux 安装入门

## 1 安装系统

vmware，一路安装，选择磁盘为单个文件，立刻分配。并且将虚拟机和磁盘放在一个文件夹中方便移动。如果选择了以后装系统，选择编辑设置，设置cd/dvd，自动连接，选择ios镜像，然后就像平常一样装系统。如果出现打印机错误，则编辑设置移除打印机，如果出现无法连接SATA0，则设置移除cd/dvd，。关闭虚拟机时先关系统再关虚拟机，挂起则将内存保存于硬盘上，下次启动可以快速进入。关机则关闭系统退出。

网络NAT模式和桥接模式的区别：NAT是VMware虚拟出交换机让windows和虚拟机通信。这是主从的模式。而桥接模式则是网卡上分配两个多个ip，用ifconfig eth0 up/down开启关闭网卡。是可以通过局域网互相访问电脑或者访问虚拟机，因此虚拟机和主机有同等地位，则ip地址不能相同。

## 2 设置超级用户权限。

ctrl+alt/ ctrl+g鼠标脱离虚拟机/进入虚拟机, ctrl+alt+T进入命题提示符。显示“$”为普通用户,#为超级用户。输入sudo passwd ，输入初始登录密码，输入超级用户密码，输入su或su root切换为超级用户“#”。输入su user切换为普通用户。要更改超级密码的方式也是一样的。

## 3 安装软件

命令格式：$sudo apt install 软件名或者#apt install 软件名

第一步，安装vim,输入vim如果有则不用装，装好后再输入vim检查是否装好。

第二步，安装ssh，然后用ps –elf|grep ssh 来检测出现sshd-D则成功。

第三步，用ifconfig命令记录下本机ip 192.168.220.130

第四步，打开PUTTY.exe 输入ip后,windows->appearance->change，改变字体，然后选择color，设置using system color，然后输入名字save->open。输入ubantu的账号和登录密码。

第五步，安装chrome

输入wget <https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb>

输入 sudo dpkg -i google-chrome-stable\_current\_amd64.deb

输入google-chrome-stable

第六步，设置固定ip，点击网络连接，编辑ipv4，新增，输入ifconfig查询到的的ip 192.168.220.130，route 查看网关地址。输入掩码，网关地址，主机地址。即可。如果需要连接其他网络则可增加一个自动获取ip的网络连接即可以。

## **4 修改显示分辨率**

在超级终端下输入如下命令行：

（1）首先使用 xrandr 命令列出当前所能检测到的分辨率，如一台显示器，最佳分辨率为 1440x900，由于未能正确检测，使用 xrandr 命令结果如下：

Screen 0: minimum 1 x 1, current 1596 x 747, maximum 16384 x 16384

Virtual1 connected primary 1596x747+0+0 (normal left inverted right x axis y axis) 0mm x 0mm

1596x747 60.00\*+

2560x1600 59.99

1920x1440 60.00

1856x1392 60.00

1792x1344 60.00

1920x1200 59.88

1600x1200 60.00

1680x1050 59.95

1400x1050 59.98

1280x1024 60.02

1440x900 59.89

1280x960 60.00

1360x768 60.02

1280x800 59.81

1152x864 75.00

1280x768 59.87

1024x768 60.00

800x600 60.32

640x480 59.94

1600x900 59.95

Virtual2 disconnected (normal left inverted right x axis y axis)

Virtual3 disconnected (normal left inverted right x axis y axis)

Virtual4 disconnected (normal left inverted right x axis y axis)

Virtual5 disconnected (normal left inverted right x axis y axis)

Virtual6 disconnected (normal left inverted right x axis y axis)

Virtual7 disconnected (normal left inverted right x axis y axis)

Virtual8 disconnected (normal left inverted right x axis y axis)

可以看到当前使用的显卡是 Virtual1（后续显示模式在此显卡上添加），列表中并无 1600x900 的分别率可用。

（2）这时可以使用 xrandr 命令新增显示模式，由于显示模式有一定的格式，如果不知道如何配置，可以通过cvt 来获得。cvt 使用方法：

$ cvt X Y // X 表示分辨率宽度，Y 表示分辨率高度

如生成 1600x900 的分辨率，可使用命令：cvt 1600 900 ,默认刷新率为 60.0Hz。生成以下模式：

# 1600x900 59.95 Hz (CVT 1.44M9) hsync: 55.99 kHz; pclk: 118.25 MHz

Modeline "1600x900\_60.00" 118.25 1600 1696 1856 2112 900 903 908 934 -hsync+vsync

（4）将 cvt 得到的显示模式使用 xrandr 命令添加：

sudo xrandr --newmode "1600x900" 118.25 1600 1696 1856 2112 900 903 908 934 -hsync +vsync

sudo xrandr --addmode Virtual1 1600x900

sudo xrandr --output Virtual1 --mode 1600x900

（5）这时就可以在显示器分辨率设置里使用 1600x900 的分辨率了。

（6）可能在每次开机时，又不能使用设定好的分辨率了，使用如下命令：

sudo gedit ~/.profile

在 ~/.profile 最末尾添加修改分辨率的命令：复制代码

cvt 1600 900

xrandr --newmode "1600x900" 118.25 1600 1696 1856 2112 900 903 908 934 -hsync +vsync

xrandr --addmode Virtual1 1600x900

重启系统后再登录即可，此时再用 xrandr 命令，即可看到新添加的分辨率。

## 5 安装VMware tools

第一步：选择菜单栏VM->setting 下，选中CD/DVD，右边点选Use ISO image file，选择镜像路径为VMware安装文件夹下的Linux.iso

第二步：选择菜单栏VM->install VMware tools下，会弹出VMware tools文件夹，里面有一个压缩包VMwareTools-8.4.5-324285.tar.gz

第三部：在桌面home文件夹里新建一个temp文件夹，将压缩包右键copy到这里来

第四步：新建终端，输入cd temp，tar xzvf VMwareTools-8.4.5-324285.tar.gz（版本号可能不一样，输入自己的版本号），解压缩好后输入cd vmware-tools-distrib，然后用root用户运行文件，即输入 sudo ./vmware-install.pl 。

第五步：开始安装了，没有yes提示时，按回车，有yes提示，输入yes。第一次yes处，等的时间会比较久，不要以为是卡住了

第六步：直到出现enjoy vmware

第七步：OK，安装好了，要重启下，输入sudo reboot

第八步：安装好后，就可以在虚拟机和window间自由移动鼠标，而且虚拟机里鼠标移动更流畅了。

第九步：此时，打开VM->setting 里，选Options下的Shared Folders ，点选always enable，在下面Add..添加你想和window硬盘共享的一个文件夹，共享的文件夹在filesystem mnt/hgfs/下找到。

## 6 命令行提示符颜色设置

用户在进行配置之前必须先了解各个文件的作用时机，对终端的配置，可以写到不同的文件之中，linux启动后环境变量加载的顺序为：etc/profile → /etc/profile.d/\*.sh → ~/.bash\_profile → ~/.bashrc → [/etc/bashrc]：

/etc/profile:此文件为系统的每个用户设置环境信息,当用户第一次登录时,该文件被执行.并从/etc /profile.d目录的配置文件中搜集shell的设置.此文件默认调用/etc/bash.bashrc文件。

/etc/bashrc:为每一个运行bash shell的用户执行此文件.当bash shell被打开时,该文件被读取.

~/.bash\_profile:每个用户都可使用该文件输入专用于自己使用的shell信息,当用户登录时, 该文件仅仅执行一次!默认情况下,他设置一些环境变量,执行用户的.bashrc文件.（放在用户主目录）

~/.bashrc:该文件包含专用于你的bash shell的bash信息。

~/.bash\_logout:当每次退出系统(退出bash shell)时,执行该文件.

针对于用户的配置，应该将配置信息写入~/.bashrc文件。想修改某用户登录后shell字体的颜色，可在~/.bashrc中添加PS1内容即可：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 前景 | 背景 | 颜色 | 序列 | 意义 |
| 30 | 40 | 红 | \! | 当前命令在历史缓冲区中的位置 |
| 31 | 41 | 绿 | **\#** | 命令编号（只要您键入内容，它就会在每次提示时累加） |
| 32 | 42 | 黄 | **\$** | 普通用户 "$"；超级用户 "#" |
| 33 | 43 | 蓝 | **\xxx** | 一个用三位数 xxx（用零代替未使用的数字， \007）表示的 ASCII 字符 |
| 34 | 44 | 紫红 | \[ | 出现在不移动光标的字符序列之前。使bash能够正确计算自动换行。 |
| 35 | 45 | 青蓝 | \\ | 反斜杠 |
| 36 | 46 | 白 | \] | 这个序列应该出现在非打印字符序列之后。 |
| 37 | 47 | 透明 | \ | ASCII转义字符（也可以键入 \033） |
| 代码 | 颜色 | | \a | ASCII响铃字符（\007） |
| 0 | OFF | | \d | "Wed Sep 06"格式的日期 |
| 1 | 高亮 | | \h | 主机名的第一部分 |
| 4 | 下划线 | | \H | 主机的全称 |
| 5 | 闪烁 | | \j | 在此shell中通过按 ^Z挂起的进程数 |
| 7 | 反转 | | **\l** | 此 shell的终端设备名 |
| 8 | 不可见 | | **\@** | 带有 am/pm的 12小时制时间 |
| 序列 | 意义 | | \V | Bash版本（包括补丁级别） |
| **\n** | 换行 符 | | \w | 当前工作目录包括根目录 |
| **\r** | 回车 符 | | \W | 当前工作目录 |
| **\u** | 用户名 | | **\t** | 24小时制时间 |
| **\v** | bash的版本 | | **\T** | 12小时制时间 |
| **\s** | shell的名称 | |  |  |

在对以上知识有了基本了解之后，我们就可以进行实际操作了。首先在终端中使用命令“cp ~/.bashrc ~/.bashrc.backup”对配置文件进行备份（这里提醒各位，在Linux中，如果需要对配置文件之类的东西进行修改的话，最好进行备份，以防 出错之后导致系统无法登录之类的问题），然后“gedit ~/.bashrc"打开文件或者用vim ~/.bashrc，找到PS1变量设置的代码，注释之（在每一行代码前插入注释符号”#“）。在文件的末尾加上”PS1='\u@\h:\w\$ '“(这是终端的默认设置，即不加任何颜色):

完成以上操作后保存并退出GEdit编辑器，回到终端后输入“source ~/.bashrc”，发现显示用户信息的地方没什么变化，说明刚才的设置与默认终端设置一致。而接下来，我们就需要对显示的信息的字体颜色进行设置。

### 6.1 设置颜色

颜色： \033[代码;前景;背景m

如：\033[1;32;40m 表示高亮显示字体为绿色，背景色为黑色

注意：

要将全部非打印字符用专用的 bash 转义序列 "\["和 "\]"括起来。这两个序列通知bash，被括起来的字符不占用行上的任何空间，这样就使自动换行能够继续正常工作。这点很关键要不然等你设好了颜色，你的终端换行会乱七八糟

现在对上边的PS1变量进行颜色的设置，即在需要更改颜色的地方插入颜色值，比如我们让用户名以红色显示的命令应该是：

PS1='\[\033[1;31;40m\]\u@\h:\w\$ '

事实上我们并不是想要所有信息的显示颜色都一样，所以我们要想办法让信息显示以不同颜色展示。这里大家可能会注意到，颜色设置的作用范围是从当前设置命令 到下一颜色设置命令的，也就是说，如果我们只在\u前设置了颜色，那么终端的字体应该都是红色的，就像下边一样，不仅信息项是红色的，就连输入命令都是红色的,为此，我们需要在@符号前面插入新的颜色值以防止整个信息项以红色显示，修改后的PS1变量为：

PS1='\[\033[1;31;40m\]\u\[\033[00m\]@\h:\w\$ '

明白了对PS1变量进行设置的原理之后，用户就可以根据自己的需要随心所欲地设置PS1变量了，比如在PS1变量中插入字符串，或者为每个信息项设置不同 的颜色，或者控制信息栏显示的内容：

我们可以让用户名带下划线，路径以白色显示，并且美元符号以绿色显示，命令则以蓝色显示：

PS1='\[\033[4;31;40m\]\u\[\033[00m\]@\h:\[\033[37;40m\]\w\[\033[32;40m\]\$ \[\033[34;40m\]'

### 6.2 命令提示符的长路径名问题

使终端只显示用户当前的工作目录的名字的方法

在PS1变量设置终端只显示工作目录的基名，即将PS1变量中的\w的小写的w换 成大写的W，修改后的PS1变量为：

PS1='\[\033[4;31;40m\]\u\[\033[00m\]@\h:\[\033[37;40m\]\W\[\033[32;40m\]\$ \[\033[34;40m\]'

可以发现，经过修改PS1变量的值之后，终端将只显示当前工作目录的基名了～。

最后，可能各位还会发现，如果打开新的终端，用户定义的配置并没有显示出来，而还是显示默认的配置。 这个时候可以通过source ~/.bashrc 命令去手动加载文件，这时候，我们所设置的配置又回来了。

当然，终端作为我们在系统之中常用的工具之一，如果每次打开终端都要source一次，那岂不是蛋疼死了？回到文章开头的地方，我们说/etc/profile文件从 /etc/profile.d目录的配置文件中搜集shell的设置.此文件默认调用/etc/bashrc文件，于是我们可以打开profile文件看一看：

$sudo gedit /etc/bashrc”(注意这里需要加上sudo!)

可以看到，在文件中有一句调用/etc/bash.bashrc文件的代码，所以可以推断，前文所说的profile文件自动加载bash.bashrc 文件,事实上就是由代码实现的，于是我们同样地可以在这里添加一句新的代码，以使profile自动加载用户的.bashrc文件，以此实现我们不需每次 打开终端都要source的需求。

保存之后关闭编辑器，新打开一个终端，可以发现，我们之前的配置已经能够自动加载了。

### 6.3 ls命令设置

在终端一个常用的命令就是ls命令了，我们可以通过在.bashrc文件中设置alias以实现ls命令的一些自定义设置：

alias其实是相当于一个命令引用的东西，例如最后一句，每当你在终端里边输入ls命令，终端会根据.bashrc文件中的alias的设置查找到引用 的命令：ls --color=auto，所以你每次输入“ls”其实就相当于输入了“ls --color=auto”，而这句命令的作用是显示当前目录的可见文件，并且以颜色区分不同类型的文件，比如文件夹和普通文件。

最后一个值得一提的就是一个小建议了，之前看到的我的终端都是黑底的，其实用户可以直接在终端中设置首选项以实现半透明背景，或者是以一张图片作为背景也可以

### 6.4 设置解释

PS1="\[\e[37;40m\][\[\e[32;40m\]\u\[\e[37;40m\]@\h\[\e[35;40m\]\W\[\e[0m\]]\\$ "

PS1="\[\033[1;32;40m\][\[\033[0;32;40m\]\u@\h:\[\033[1;35;40m\]\w\[\033[1;32;40m\]]\[\033[1;31;40m\]\$\[\033[1;32;40m\] "

" ------------>起始引号

\[\033[1;32;40m\][ ------------>定义最右边的" [ " ；

1：字体宽度，好像有加亮功能.

32：字符颜色。32表示绿色。

40：背景色，40表示黑色。

\[\033[0;32;40m\]\u@\h: --------->定义右边的 \u@\h: 这几个字符颜色。

0： 默认字体宽度。

32：字符颜色。绿色。

40: 背景颜色。黑色。

\[\033[1;35;40m\]\w ---------> 定义右边的 \w (路径颜色) 。

1：字符宽度。加亮。

35：字符颜色。洋红。

40：背景色。黑色。

\[\033[1;32;40m\]] ---------> 定义右边的 ] 字符。跟前面的 [字符使用同一效果。

\[\033[1;31;40m\]\$ ---------> 定义了$字符为红色。

\[\033[1;32;40m \] -----------> 这个定义了你在命令行要输入命令显示的颜色。

" ------------>结束引号

如果果不想改变输入颜色的话，用\[\033[0m\] 关闭转义序列， \033[0m 是 shell 的默认颜色。按照上面的例子来说就是在"\[\033[0m\] 前面加上即可。

## 7 改变安装源

设置，软件更新选择阿里云，reload即可。

## 8 vim更新

Ubuntu已经为我们提供了ppa源来安装

sudo add-apt-repository ppa:jonathonf/vim

sudo apt update

sudo apt install vim

如果您想要卸载它, 请使用如下命令

sudo apt remove vim

sudo add-apt-repository --remove ppa:jonathonf/vim

## 9 vim实现显示行号和括号自动补齐对齐

vim的设置文件都在/etc/vim/vimrc中或者/home/用户名/.vimrc中，要使用超级权限才可以修改。改好久生效了。

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
" 显示相关    
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
"set shortmess=atI   " 启动的时候不显示那个援助乌干达儿童的提示 (如果报错可以注释了)   
"winpos 5 5          " 设定窗口位置    
"set lines=40 columns=155    " 设定窗口大小    
"set number              " 显示行号    
set go=             " 不要图形按钮    
"color asmanian2     " 设置背景主题    
set guifont=Courier\_New:h10:cANSI   " 设置字体    
syntax on           " 语法高亮    
autocmd InsertLeave \* se nocul  " 用浅色高亮当前行    
autocmd InsertEnter \* se cul    " 用浅色高亮当前行    
"set ruler           " 显示标尺    
set showcmd         " 输入的命令显示出来，看的清楚些    
"set cmdheight=1     " 命令行（在状态行下）的高度，设置为1    
"set whichwrap+=<,>,h,l   " 允许backspace和光标键跨越行边界(不建议)    
"set scrolloff=3     " 光标移动到buffer的顶部和底部时保持3行距离    
set novisualbell    " 不要闪烁(不明白)    
set statusline=%F%m%r%h%w\ [FORMAT=%{&ff}]\ [TYPE=%Y]\ [POS=%l,%v][%p%%]\ %{strftime(\"%d/%m/%y\ -\ %H:%M\")}   "状态行显示的内容    
set laststatus=1    " 启动显示状态行(1),总是显示状态行(2)    
set foldenable      " 允许折叠    
set foldmethod=manual   " 手动折叠    
"set background=dark "背景使用黑色   
set nocompatible  "去掉讨厌的有关vi一致性模式，避免以前版本的一些bug和局限    
" 显示中文帮助  
if version >= 603  
    set helplang=cn  
    set encoding=utf-8  
endif  
" 设置配色方案  
"colorscheme murphy  
"字体   
"if (has("gui\_running"))   
"   set guifont=Bitstream\ Vera\ Sans\ Mono\ 10   
"endif   
set fencs=utf-8,ucs-bom,shift-jis,gb18030,gbk,gb2312,cp936  
set termencoding=utf-8  
set encoding=utf-8  
set fileencodings=ucs-bom,utf-8,cp936  
set fileencoding=utf-8  
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
"""""新文件标题""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
"新建.c,.h,.sh,.java文件，自动插入文件头   
autocmd BufNewFile \*.cpp,\*.[ch],\*.sh,\*.java exec ":call SetTitle()"   
""定义函数SetTitle，自动插入文件头   
func SetTitle()   
    "如果文件类型为.sh文件   
    if &filetype == 'sh'   
        call setline(1,"\#########################################################################")   
        call append(line("."), "\# File Name: ".expand("%"))   
        call append(line(".")+1, "\# Author: ma6174")   
        call append(line(".")+2, "\# mail: ma6174@163.com")   
        call append(line(".")+3, "\# Created Time: ".strftime("%c"))   
        call append(line(".")+4, "\#########################################################################")   
        call append(line(".")+5, "\#!/bin/bash")   
        call append(line(".")+6, "")   
    else   
        call setline(1, "/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")   
        call append(line("."), "    > File Name: ".expand("%"))   
        call append(line(".")+1, "    > Author: liao20081228")   
        call append(line(".")+2, "    > Mail: liao20081228@126.com ")   
        call append(line(".")+3, "    > Created Time: ".strftime("%c"))   
        call append(line(".")+4, " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/")   
        call append(line(".")+5, "")  
    endif  
    if &filetype == 'cpp'  
        call append(line(".")+6, "#include<iostream>")  
        call append(line(".")+7, "using namespace std;")  
        call append(line(".")+8, "")  
    endif  
    if &filetype == 'c'  
        call append(line(".")+6, "#include<stdio.h>")  
 call append(line(".")+6, "#include<stdlib.h>")  
        call append(line(".")+7, "")  
    endif  
    "新建文件后，自动定位到文件末尾  
    autocmd BufNewFile \* normal G  
endfunc   
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
"键盘命令  
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
  
nmap <leader>w :w!<cr>  
nmap <leader>f :find<cr>  
  
" 映射全选+复制 ctrl+a  
map <C-A> ggVGY  
map! <C-A> <Esc>ggVGY  
map <F12> gg=G  
" 选中状态下 Ctrl+c 复制  
vmap <C-c> "+y  
"去空行    
nnoremap <F2> :g/^\s\*$/d<CR>   
"比较文件    
nnoremap <C-F2> :vert diffsplit   
"新建标签    
map <M-F2> :tabnew<CR>    
"列出当前目录文件    
map <F3> :tabnew .<CR>    
"打开树状文件目录    
map <C-F3> \be    
"C，C++ 按F5编译运行  
map <F5> :call CompileRunGcc()<CR>  
func! CompileRunGcc()  
    exec "w"  
    if &filetype == 'c'  
        exec "!g++ % -o %<"  
        exec "! ./%<"  
    elseif &filetype == 'cpp'  
        exec "!g++ % -o %<"  
        exec "! ./%<"  
    elseif &filetype == 'java'   
        exec "!javac %"   
        exec "!java %<"  
    elseif &filetype == 'sh'  
        :!./%  
    endif  
endfunc  
"C,C++的调试  
map <F8> :call Rungdb()<CR>  
func! Rungdb()  
    exec "w"  
    exec "!g++ % -g -o %<"  
    exec "!gdb ./%<"  
endfunc  
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
""实用设置  
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
" 设置当文件被改动时自动载入  
set autoread  
" quickfix模式  
autocmd FileType c,cpp map <buffer> <leader><space> :w<cr>:make<cr>  
"代码补全

set completeopt=preview,menu   
"允许插件    
filetype plugin on  
"共享剪贴板    
set clipboard+=unnamed   
"从不备份    
set nobackup  
"make 运行  
:set makeprg=g++\ -Wall\ \ %  
"自动保存  
set autowrite  
set ruler                   " 打开状态栏标尺  
set cursorline              " 突出显示当前行  
set magic                   " 设置魔术  
set guioptions-=T           " 隐藏工具栏  
set guioptions-=m           " 隐藏菜单栏  
"set statusline=\ %<%F[%1\*%M%\*%n%R%H]%=\ %y\ %0(%{&fileformat}\ %{&encoding}\ %c:%l/%L%)\  
" 设置在状态行显示的信息  
set foldcolumn=0  
set foldmethod=indent " 自动折叠  
set foldlevel=3   
set foldenable              " 允许折叠  
" 不要使用vi的键盘模式，而是vim自己的  
set nocompatible  
" 语法高亮  
set syntax=on  
" 去掉输入错误的提示声音  
set noeb  
" 在处理未保存或只读文件的时候，弹出确认  
set confirm  
" 自动缩进  
set autoindent " 自动对齐（继承前一行的缩进方式）  
set cindent  
" Tab键的宽度  
set tabstop=4  
" 统一缩进为4  
set softtabstop=4  
set shiftwidth=4  
" 不要用空格代替制表符  
set noexpandtab  
" 在行和段开始处使用制表符  
set smarttab  
" 显示行号  
set number  
" 历史记录数  
set history=1000  
"禁止生成临时文件  
set nobackup  
set noswapfile  
"搜索忽略大小写  
set ignorecase  
"搜索逐字符高亮  
set hlsearch  
set incsearch  
"行内替换  
set gdefault  
"编码设置  
set enc=utf-8  
set fencs=utf-8,ucs-bom,shift-jis,gb18030,gbk,gb2312,cp936  
"语言设置  
set langmenu=zh\_CN.UTF-8  
set helplang=cn  
" 我的状态行显示的内容（包括文件类型和解码）  
"set statusline=%F%m%r%h%w\ [FORMAT=%{&ff}]\ [TYPE=%Y]\ [POS=%l,%v][%p%%]\ %{strftime(\"%d/%m/%y\ -\ %H:%M\")}  
"set statusline=[%F]%y%r%m%\*%=[Line:%l/%L,Column:%c][%p%%]  
" 总是显示状态行  
set laststatus=2  
" 命令行（在状态行下）的高度，默认为1，这里是2  
set cmdheight=2  
" 侦测文件类型  
filetype on  
" 载入文件类型插件  
filetype plugin on  
" 为特定文件类型载入相关缩进文件  
filetype indent on  
" 保存全局变量  
set viminfo+=!  
" 带有如下符号的单词不要被换行分割  
set iskeyword+=\_,$,@,%,#,-  
" 字符间插入的像素行数目  
set linespace=0  
" 增强模式中的命令行自动完成操作  
set wildmenu  
" 使回格键（backspace）正常处理indent, eol, start等  
set backspace=2  
" 允许backspace和光标键跨越行边界  
set whichwrap+=<,>,h,l  
" 可以在buffer的任何地方使用鼠标（类似office中在工作区双击鼠标定位）  
set mouse=a  
set selection=exclusive  
set selectmode=mouse,key  
" 通过使用: commands命令，告诉我们文件的哪一行被改变过  
set report=0  
" 在被分割的窗口间显示空白，便于阅读  
set fillchars=vert:\ ,stl:\ ,stlnc:\  
" 高亮显示匹配的括号  
set showmatch  
" 匹配括号高亮的时间（单位是十分之一秒）  
set matchtime=1  
" 光标移动到buffer的顶部和底部时保持3行距离  
set scrolloff=3  
"  智能自动缩进（以c程序的方式）  
set smartindent  
" 高亮显示普通txt文件（需要txt.vim脚本）  
au BufRead,BufNewFile \*  setfiletype txt  
"自动匹配括号版本1(按左右括号再按右括号 自动后移俩个字符，不会重新输入右括号)  
inoremap ( ()<ESC>i  
inoremap [ []<ESC>i  
inoremap { {<CR>}<ESC>O   
autocmd Syntax html,vim inoremap < <lt>><Esc>i| inoremap > <c-r>=ClosePair('>')<CR>   
inoremap ) <c-r>=ClosePair(')')<CR>  
inoremap ] <c-r>=ClosePair(']')<CR>  
inoremap } <c-r>=CloseBracket()<CR>  
inoremap " <c-r>=QuoteDelim('"')<CR>  
inoremap ' <c-r>=QuoteDelim("'")<CR>  
function ClosePair(char)  
 if getline('.')[col('.') - 1] == a:char  
 return "\<Right>"  
 else  
 return a:char  
 endif  
endfunction  
function CloseBracket()  
 if match(getline(line('.') + 1), '\s\*}') < 0  
 return "\<CR>}"  
 else  
 return "\<Esc>j0f}a"  
endif  
endfunction  
   
function QuoteDelim(char)  
 let line = getline('.')  
 let col = col('.')  
 if line[col - 2] == "\\"  
"Inserting a quoted quotation mark into the string   
 return a:char  
 elseif line[col - 1] == a:char  
"Escaping out of the string   
 return "\<Right>"   
 else   
"Starting a string   
 return a:char.a:char."\<Esc>i"   
 endif   
endfunction   
"自动匹配括号版本2(按左右括号再按右括号 不会后移俩个字符，会再生成一个右括号)  
inoremap ( ()<ESC>i  
inoremap ) <c-r>=ClosePair(')')<CR>  
inoremap { {<CR>}<ESC>O  
inoremap } <c-r>=ClosePair('}')<CR>  
inoremap [ []<ESC>i  
inoremap ] <c-r>=ClosePair(']')<CR>  
inoremap " ""<ESC>i  
inoremap ' ''<ESC>i  
function! ClosePair(char)  
    if getline('.')[col('.') - 1] == a:char  
        return "\<Right>"  
    else  
        return a:char  
    endif  
endfunction  
filetype plugin indent on   
"打开文件类型检测, 加了这句才可以用智能补全  
set completeopt=longest,menu  
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
" CTags的设定    
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
let Tlist\_Sort\_Type = "name"    " 按照名称排序    
let Tlist\_Use\_Right\_Window = 1  " 在右侧显示窗口    
let Tlist\_Compart\_Format = 1    " 压缩方式    
let Tlist\_Exist\_OnlyWindow = 1  " 如果只有一个buffer，kill窗口也kill掉buffer    
let Tlist\_File\_Fold\_Auto\_Close = 0  " 不要关闭其他文件的tags    
let Tlist\_Enable\_Fold\_Column = 0    " 不要显示折叠树    
autocmd FileType java set tags+=D:\tools\java\tags    
"autocmd FileType h,cpp,cc,c set tags+=D:\tools\cpp\tags    
"let Tlist\_Show\_One\_File=1            "不同时显示多个文件的tag，只显示当前文件的  
"设置tags    
set tags=tags    
"set autochdir   
  
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
"其他东东  
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""  
"默认打开Taglist   
let Tlist\_Auto\_Open=1   
""""""""""""""""""""""""""""""   
" Tag list (ctags)   
""""""""""""""""""""""""""""""""   
let Tlist\_Ctags\_Cmd = '/usr/bin/ctags'   
let Tlist\_Show\_One\_File = 1 "不同时显示多个文件的tag，只显示当前文件的   
let Tlist\_Exit\_OnlyWindow = 1 "如果taglist窗口是最后一个窗口，则退出vim   
let Tlist\_Use\_Right\_Window = 1 "在右侧窗口中显示taglist窗口  
" minibufexpl插件的一般设置  
let g:miniBufExplMapWindowNavVim = 1  
let g:miniBufExplMapWindowNavArrows = 1  
let g:miniBufExplMapCTabSwitchBufs = 1  
let g:miniBufExplModSelTarget = 1

**解释文件3**

set nocompatible "不要使用vi的键盘模式，而是vim自己的

source $VIMRUNTIME/mswin.vim

behave mswin "兼容windows下的快捷键

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" GVIM自身的设置

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

language messages zh\_CN.utf-8 " 解决consle输出乱码

colorscheme desert " 灰褐色主题

set guioptions-=T " 隐藏工具栏

set guifont=Monaco:h10 " 字体 && 字号

set noerrorbells " 关闭错误提示音

set nobackup " 不要备份文件

set linespace=0 " 字符间插入的像素行数目

set shortmess=atI " 启动的时候不显示那个援助索马里儿童的提示

set novisualbell " 不要闪烁

set scrolloff=3 " 光标移动到buffer的顶部和底部时保持3行距离

set mouse=a " 可以在buffer的任何地方 ->

set selection=exclusive " 使用鼠标（类似office中 ->

set selectmode=mouse,key " 在工作区双击鼠标定位）

set cursorline " 突出显示当前行

set nu! " 显示行号

set whichwrap+=<,>,h,l " 允许backspace和光标键跨越行边界

set completeopt=longest,menu "按Ctrl+N进行代码补全

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" 文本格式和排版

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

set list " 显示Tab符，->

set listchars=tab:\|\ , " 使用一高亮竖线代替

set tabstop=4 " 制表符为4

set autoindent " 自动对齐（继承前一行的缩进方式）

set smartindent " 智能自动缩进（以c程序的方式）

set softtabstop=4

set shiftwidth=4 " 换行时行间交错使用4个空格

set noexpandtab " 不要用空格代替制表符

set cindent " 使用C样式的缩进

set smarttab " 在行和段开始处使用制表符

set nowrap " 不要换行显示一行

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" 状态行(命令行)的显示

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

set cmdheight=2 " 命令行（在状态行下）的高度，默认为1，这里是2

set ruler " 右下角显示光标位置的状态行

set laststatus=2 " 开启状态栏信息

set wildmenu " 增强模式中的命令行自动完成操作

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" 文件相关

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

set fenc=utf-8

set encoding=utf-8 " 设置vim的工作编码为utf-8，如果源文件不是此编码，vim会进行转换后显示

set fileencoding=utf-8 " 让vim新建文件和保存文件使用utf-8编码

set fileencodings=utf-8,gbk,cp936,latin-1

filetype on " 侦测文件类型

filetype indent on " 针对不同的文件类型采用不同的缩进格式

filetype plugin on " 针对不同的文件类型加载对应的插件

syntax on " 语法高亮

filetype plugin indent on " 启用自动补全

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" 查找

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

set hlsearch " 开启高亮显示结果

set nowrapscan " 搜索到文件两端时不重新搜索

set incsearch " 开启实时搜索功能

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" 语言的编译和运行

" 支持的语言：java F5编译(保存+编译) F6运行

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

func! CompileCode()

exec "w"

if &filetype == "java"

exec "!javac -encoding utf-8 %"

endif

endfunc

func! RunCode()

if &filetype == "java"

exec "!java -classpath %:h; %:t:r"

endif

endfunc

" F5 保存+编译

map <F5> :call CompileCode()<CR>

" F6 运行

map <F6> :call RunCode()<CR>

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" 实用功能

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"--------引号 && 括号自动匹配

:inoremap ( ()<ESC>i

:inoremap ) <c-r>=ClosePair(')')<CR>

:inoremap { {}<ESC>i

:inoremap } <c-r>=ClosePair('}')<CR>

:inoremap [ []<ESC>i

:inoremap ] <c-r>=ClosePair(']')<CR>

":inoremap < <><ESC>i

":inoremap > <c-r>=ClosePair('>')<CR>

:inoremap " ""<ESC>i

:inoremap ' ''<ESC>i

:inoremap ` ``<ESC>i

function ClosePair(char)

if getline('.')[col('.') - 1] == a:char

return "\<Right>"

else

return a:char

endif

endf

"--------启用代码折叠，用空格键来开关折叠

set foldenable " 打开代码折叠

set foldmethod=syntax " 选择代码折叠类型 syntax语法,manual 手动折叠

set foldcolumn=0 " 设置折叠区域的宽度

setlocal foldlevel=1 " 设置折叠层数为(setlocal)

set foldlevel=100 " 禁止自动折叠

" set foldclose=all " 设置为自动关闭折叠

nnoremap <space> @=((foldclosed(line('.')) < 0) ? 'zc':'zo')<CR>

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" 插件

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" <F9>打开文件浏览窗口 插件为WinManager

let g:winManagerWindowLayout='FileExplorer'

nmap <F9> :WMToggle<CR>

" MiniBufExplorer

let g:miniBufExplMapWindowNavVim = 1

let g:miniBufExplMapWindowNavArrows = 1

let g:miniBufExplMapCTabSwitchBufs = 1

let g:miniBufExplModSelTarget = 1