# 软件安装

## Ubuntu专区

### gcc g++ gdb

#### Ubuntu 18.04 LTS默认联网安装

|  |
| --- |
| sudo apt install gcc  sudo apt install gcc-c++  sudo apt install gdb |

#### Ubuntu 18.04 LTS在线安装gcc9 g++9

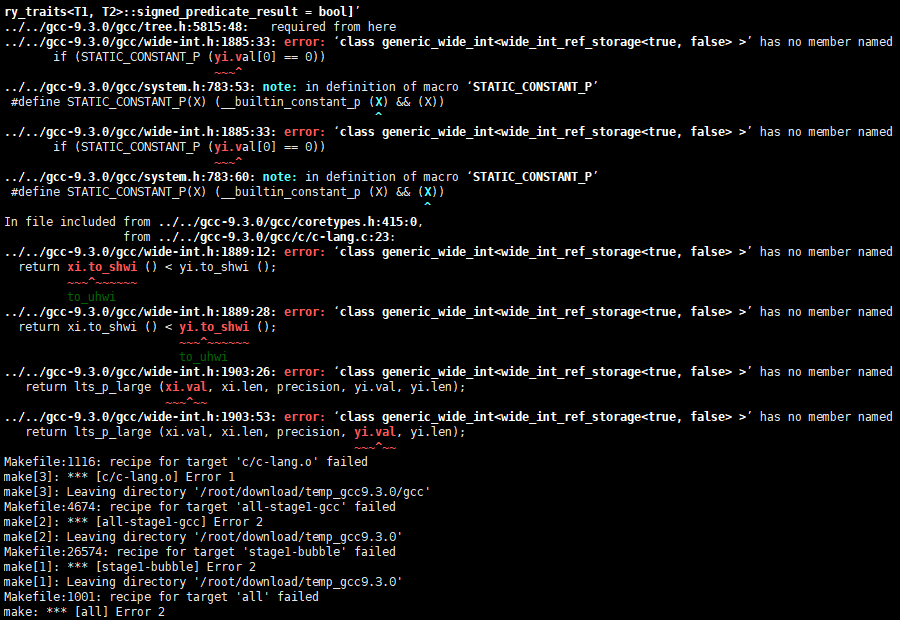
现在可以在[ubuntu-toolchain-r](https://launchpad.net/~ubuntu-toolchain-r/+archive/ubuntu/test)中使用它：

|  |
| --- |
| sudo add-apt-repository ppa:ubuntu-toolchain-r/test  sudo apt update  sudo apt install gcc-9  sudo apt install g++-9 |

若因为网络问题安装ppa:ubuntu-toolchain-r/test错误，参看：2.1.3.2 安装proxychains，让终端命令使用socks5/http代理

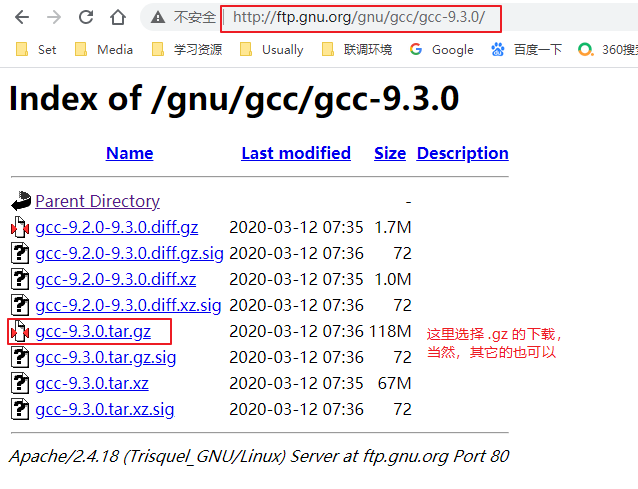
#### Ubuntu 18.04 LTS离线源码安装gcc 9.3.0

**注意：**现有gcc 7.5.0，升级gcc 9.3.0失败，这里仅仅记录步骤作为参考。第7步编译出错：



**安装步骤：**

1、下载gcc 9.3.0安装包，网址：<http://ftp.gnu.org/gnu/gcc/gcc-9.3.0/gcc-9.3.0.tar.gz>



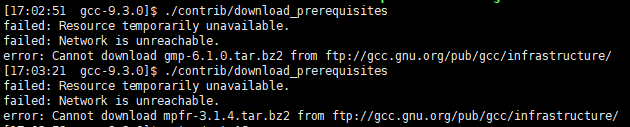
2、拷贝到linux并解压

|  |
| --- |
| cp gcc-9.3.0.tar.gz /root/download  tar xzvf gcc-9.3.0.tar.gz /root/download #解压时间比较长，下载的.gz包大概100M+，解压后有500M+ |

3、在线下载依赖包

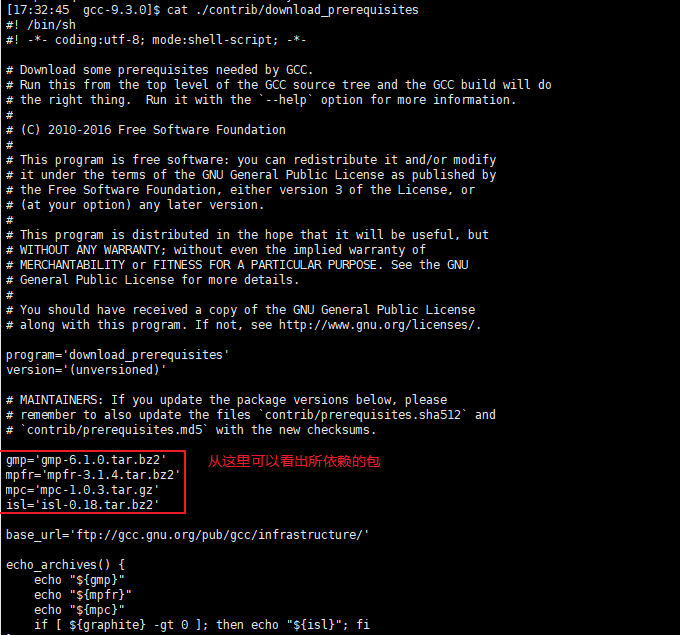
|  |
| --- |
| cd /root/download/gcc-9.3.0 #切换到目录  ./contrib/download\_prerequisites #下载依赖包 |

如果下载出错，看步骤4

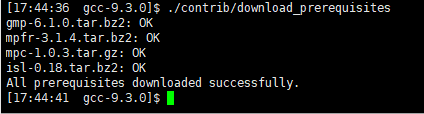


4、离线下载依赖包

运行*cat ./contrib/download\_prerequisites*，显示*download\_prerequisites*的内容，从中可以找到所需的依赖包：



从网上下载这四个依赖包（百度输入包名搜索下载即可，下载的时候可能需要梯子），然后把四个依赖包拷贝到*/root/download/gcc-9.3.0*目录下，然后再次运行*./contrib/download\_prerequisites*，看到如下提示，说明已经成功了。



5、准备编译目录

|  |
| --- |
| cd ..  mkdir temp\_gcc9.3.0 && cd temp\_gcc9.3.0 |

6、设置编译选项，生成make文件

|  |
| --- |
| ../gcc-9.3.0/configure --prefix=/usr/local/gcc-9.3.0 --enable-threads=posix --disable-checking --disable-multilib #允许多线程，不允许32位等选项 |

7、编译

|  |
| --- |
| make #编译时间较长，且这里生成的目录比较大，有6G+ |

8、安装

|  |
| --- |
| sudo make install |

#### GCC G++多版本切换

1、查看系统已安装的GCC版本

|  |
| --- |
| ls -l /usr/bin/gcc\* #也可能位于/usr/local/bin下  root@c98c1b416c01:/home# ls -l /usr/bin/gcc\*  lrwxrwxrwx 1 root root 21 Jan 16 22:09 /usr/bin/gcc -> /etc/alternatives/gcc  -rwxr-xr-x 1 root root 255080 Mar 20 2014 /usr/bin/gcc-4.4  -rwxr-xr-x 1 root root 915736 Oct 4 19:23 /usr/bin/gcc-5  lrwxrwxrwx 1 root root 8 Feb 11 2016 /usr/bin/gcc-ar -> gcc-ar-5  -rwxr-xr-x 1 root root 31136 Oct 4 19:23 /usr/bin/gcc-ar-5  lrwxrwxrwx 1 root root 8 Feb 11 2016 /usr/bin/gcc-nm -> gcc-nm-5  -rwxr-xr-x 1 root root 31136 Oct 4 19:23 /usr/bin/gcc-nm-5  lrwxrwxrwx 1 root root 12 Feb 11 2016 /usr/bin/gcc-ranlib -> gcc-ranlib-5  -rwxr-xr-x 1 root root 31136 Oct 4 19:23 /usr/bin/gcc-ranlib-5  -rwxr-xr-x 1 root root 2189 Dec 3 2015 /usr/bin/gccmakedep |

2、设置GCC G++各版本优先级

**语法格式：***update-alternatives --install <链接> <名称> <路径> <优先级>*

**设置GCC各版本优先级**

|  |
| --- |
| sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.4 60  sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 50 |

**设置G++各版本优先级**

|  |
| --- |
| sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.4 60  sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-5 50 |

版本号gcc-5、gcc-4.4、g++-5、g++-4.4以及优先级数值40、50可以根据自己需要更改。优先级数值范围为0到100。设置好GCC、G++各版本优先级后可进行优先级查看，以及个版本切换。

3、GCC G++各版本切换

**GCC各版本切换**

|  |
| --- |
| sudo update-alternatives --config gcc  root@c98c1b416c01:/home# update-alternatives --config gcc  There are 2 choices for the alternative gcc (providing /usr/bin/gcc).  Selection Path Priority Status  ------------------------------------------------------------  \* 0 /usr/bin/gcc-4.4 60 auto mode  1 /usr/bin/gcc-4.4 60 manual mode  2 /usr/bin/gcc-5 50 manual mode  Press <enter> to keep the current choice[\*], or type selection number: |

**G++各版本切换**

|  |
| --- |
| sudo update-alternatives --config g++  root@c98c1b416c01:/home# update-alternatives --config g++  There are 2 choices for the alternative g++ (providing /usr/bin/g++).  Selection Path Priority Status  ------------------------------------------------------------  \* 0 /usr/bin/g++-4.4 60 auto mode  1 /usr/bin/g++-4.4 60 manual mode  2 /usr/bin/g++-5 50 manual mode  Press <enter> to keep the current choice[\*], or type selection number: |

**另一种切换方法**

在*～/.bashrc*文件中增添加

|  |
| --- |
| alias gcc='/usr/bin/gcc-4.4'  alias g++='/usr/bin/g++-4.4' |

### electron-ssr

**简介：**GUI界面，推荐用这个

**安装：**

|  |
| --- |
| sudo apt install electron-ssr |

**GitHub网址：**[electron-ssr](https://github.com/qingshuisiyuan/electron-ssr-backup)

### python-pip -> shadowsocks

**简介：**命令行操作，无界面

**安装步骤：**

1、安装

|  |
| --- |
| sudo apt-get update  sudo apt-get install python-pip  sudo pip install shadowsocks |

2、更新

|  |
| --- |
| sudo pip install -U git+https://github.com/shadowsocks/shadowsocks.git@master |

3、设置配置文件

编辑配置文件

|  |
| --- |
| gedit /etc/shadowsocks/config.json #可以放在其它路径，也可以起其它名字 |

打开后写入如下内容（根据实际情况改写），保存关闭

|  |
| --- |
| {  "server":"xx",  "server\_port":xx,  "local\_address":"xx",  "local\_port":xx,  "password":"xx",  "timeout":xx,  "method":"xx"  } |

4、启动

|  |
| --- |
| sudo /usr/local/bin/sslocal -d start -v -c /etc/shadowsocks/config.json |

5、参考文献

<https://my.oschina.net/xiaohelong/blog/2209343?tdsourcetag=s_pctim_aiomsg>

<https://ywnz.com/linuxjc/2687.html>

### chromium

|  |
| --- |
| sudo snap install chromium |

### OneDrive

**GitHub网址**：[onedrive](https://github.com/skilion/onedrive/tree/master)

**安装步骤：**

1、安装依赖项

|  |
| --- |
| sudo apt install libcurl4-openssl-dev  sudo apt install libsqlite3-dev  sudo snap install --classic dmd && sudo snap install --classic dub |

2、安装OneDrive

|  |
| --- |
| git clone https://github.com/skilion/onedrive.git  cd onedrive  make  sudo make install |

3、配置同步位置和忽略文件

|  |
| --- |
| mkdir -p ~/.config/onedrive  cp ./config ~/.config/onedrive/config  gedit ~/.config/onedrive/config |

*sync\_dir*：同步路径

*skip\_file*：忽略文件

4、选择性同步

|  |
| --- |
| gedit ~/.config/onedrive/sync\_list |

编辑需要同步的文件，e.g.

|  |
| --- |
| Backup  Documents/latest\_report.docx  Work/ProjectX  notes.txt |

5、获取授权

终端输入*onedrive*，之后会给出一个链接，复制链接到浏览器，登录账号，把返回的链接复制到终端，*Enter*确定，开始同步。同步的时候终端会显示正在下载的内容，此时先不要关闭终端，同步完了再关闭。后续设置开机自启，等再开机就会在后台自动同步，不会在终端显示了。

6、开机自启

|  |
| --- |
| systemctl --user enable onedrive  systemctl --user start onedrive |

7、只要更改了配置文件，就要重新同步，使配置文件生效

重新同步：

|  |
| --- |
| onedrive --resync |

这个尽量不要用*nohup onedrive --resync > /dev/null 2>&1 &*使其在后台重新同步，好像这样重新同步不了。

等待下载完毕，终端光标不动，输入*nohup onedrive -m > /dev/null 2>&1 &*，在后台监视远程和本地更改，实时同步。

8、查看日志

|  |
| --- |
| journalctl --user-unit onedrive -f |

9、使用命令

Usage: onedrive [OPTION]...

|  |
| --- |
| no option Sync and exit  --confdir Set the directory used to store the configuration files  -d --download Only download remote changes  --logout Logout the current user  -m --monitor Keep monitoring for local and remote changes  --print-token Print the access token, useful for debugging  --resync Forget the last saved state, perform a full sync  --syncdir Set the directory used to sync the files that are synced  -v --verbose Print more details, useful for debugging  --version Print the version and exit  -h --help This help information |

### Typora

[官网](https://typora.io/)有Linux版本安装教程

### 搜狗输入法

1、打开ubuntu软件中心，搜索fcitx，然后安装fcitx和fcitx配置工具

2、搜狗输入法官网下载Linux版本安装包，用*dpkg -i <package\_name>*命令安装

3、**设置 > 区域和语言 > 管理已安装的语言 > 键盘输入法系统 > fcitx > 应用到整个系统 > 关闭 > 重启**

4、卸载ibus（可选步骤）

|  |
| --- |
| sudo apt-get remove ibus #卸载 ibus  sudo apt-get purge ibus #清除 ibus 配置  sudo apt-get remove indicator-keyboard #卸载顶部面板任务栏上的键盘指示 |

5、输入特殊字符的两种方法：

更改fcitx配置文件，快捷输入特殊字符。在*/usr/share/fcitx/pinyin/pySym.mb*或者*~/.config/fcitx/pinyin/pySym.mb*中定义需要使用的特殊符号，即可快速输入特殊字符，但是只对fcitx默认输入法有效，对搜狗输入法无效。可在 GitHub：[fcitx-config](https://github.com/alswl/fcitx-config/blob/master/README.asciidoc)查看使用介绍。更改后重启输入法生效。

自定义内容：

|  |
| --- |
| zj ⌈  yj ⌋  wjx ★  lx ♦  yx ● |

Ubuntu软件中心安装*字符映射表*，打开后左侧选择*公共*右侧就会出现特殊字符大全。

### 火狐浏览器/firefox

|  |
| --- |
| sudo apt install firefox |

### VLC

|  |
| --- |
| sudo add-apt-repository ppa:videolan/master-daily  sudo apt update  sudo apt-get install vlc qtwayland5 |

### Ubuntu Cleaner

**简介：**Ubuntu Cleaner是一个系统管理工具，其被特别设计用来移除不再使用的包、不需要的应用和清理浏览器缓存。Ubuntu Cleaner有易于使用的简单用户界面。Ubuntu Cleaner是BleachBit最好的替代品之一，BleachBit是Linux发行版上的相当好的清理工具。

|  |
| --- |
| sudo add-apt-repository ppa:gerardpuig/ppa  sudo apt-get update  sudo apt-get install ubuntu-cleaner |

### Stacer

**简介：**Stacer是一个开源的系统诊断和优化工具，使用Electron开发框架开发。它有一个优秀的用户界面，你可以清理缓存内存、启动应用、卸载不需要的应用、掌控后台系统进程。类似Ubuntu Cleaner。

|  |
| --- |
| sudo add-apt-repository ppa:oguzhaninan/stacer  sudo apt-get update  sudo apt-get install stacer |

### Neofetch

**简介：**显示桌面环境、内核版本、bash版本和你正在运行的GTK主题的信息。

|  |
| --- |
| sudo add-apt-repository ppa:dawidd0811/neofetch  sudo apt-get update  sudo apt-get update install neofetch |

### 字体管理器

|  |
| --- |
| sudo add-apt-repository ppa:font-manager/staging  sudo apt-get update  sudo apt-get install font-manager |

### PlayOnLinux

**简介：**PlayOnLinux是WINE模拟器的前端，允许你在Linux上运行Windows应用。你只需要在WINE中安装Windows应用，之后你就可以轻松的使用PlayOnLinux启动应用和游戏了。

* 依赖于：[wine:i386](https://ubuntu.pkgs.org/18.04/ubuntu-universe-i386/wine1.6-i386_1.8.4ubuntu1_i386.deb.html)
* 依赖于：xterm，安装方法：*sudo apt install xterm*
* 安装[PlayOnLinux](https://www.playonlinux.com/en/download.html)，不同版本安装方法不一样，官网有每个版本的安装方法，这里只给出Ubuntu 18.04 LTS安装方法：

|  |
| --- |
| wget -q "http://deb.playonlinux.com/public.gpg" -O- | sudo apt-key add -  sudo wget http://deb.playonlinux.com/playonlinux\_bionic.list -O /etc/apt/sources.list.d/playonlinux.list  sudo apt-get update  sudo apt-get install playonlinux |

### CopyQ

**简介：**CopyQ是一个简单但是非常有用的剪贴板管理器，它保存你的系统剪贴板内容，无论你做了什么改变，你都可以在你需要的时候搜索和恢复它。它是一个很棒的工具，支持文本、图像、HTML和其它格式。

|  |
| --- |
| sudo add-apt-repository ppa:hluk/copyq  sudo apt-get update  sudo apt-get install copyq |

### TeXstudio

**简介：**Texstudio是一个创建和编辑LaTex文件的集成写作环境。它是开源的编辑器，提供了语法高亮、集成的查看器、交互式拼写检查、代码折叠、拖放等特点。从官网和Ubuntu软件中心下载Texstudio。

### 简单天气指示器

**简介：**简单天气指示器是用Python开发的开源天气提示应用。它自动侦查你的位置，并显示你天气信息像温度，下雨的可能性，湿度，风速和可见度。

**GitHub网址：**[Ubuntu Indicator Weather](https://github.com/kasramp/UbuntuIndicatorWeather)

**安装方法：**

|  |
| --- |
| sudo add-apt-repository ppa:kasra-mp/ubuntu-indicator-weather  sudo apt-get update  sudo apt-get install indicator-weather |

### Nomacs

**简介：**Nomacs是一款开源、跨平台的图像浏览器，支持所有主流的图片格式包括RAW和psd图像。它可以浏览zip文件中的图像或者Microsoft Office文件并提取到目录。

|  |
| --- |
| sudo add-apt-repository ppa:nomacs/stable  sudo apt-get update  sudo apt-get install nomacs |

### zsh

1、安装zsh

|  |
| --- |
| sudo apt-get install zsh |

2、安装oh-my-zsh

|  |
| --- |
| sudo wget https://github.com/robbyrussell/oh-my-zsh/raw/master/tools/install.sh -O - | sh  chsh -s /bin/zsh |

3、重启生效

4、自定义指令加在*~/.zshrc*下

### gnome-screenshot

简介：系统自带截图软件

* 路径：*/usr/bin/gnome-screenshot*
* 使用命令：*man gnome-screenshot*
* 系统快捷键中添加如下快速截图命令：*gnome-screenshot -a -c*（截取选定区域并复制到剪切板）

### nixnote2

* 安装：*sudo apt install nixnote2*
* 国服授权登录页面无法输入密码：**授权登录页面点击左上角大象图标 > 会自动打开印象笔记官方主页 > 选择使用微信登录 > 用微信扫码登录 > 手机提示登录成功后关闭窗口，即便此时窗口可能是空白，什么也不提示 > 点击同步，会出现授权页面 > 授权自动开始同步**

### apt-fast

**简介：**apt-fast是apt-get的一个shell脚本包装器，通过从多连接同时下载包来提升更新及包下载速度。

|  |
| --- |
| sudo add-apt-repository ppa:apt-fast/stable  sudo apt-get update  sudo apt-get install apt-fast |

### notepadqq

**简介：**Linux版的notepad++

|  |
| --- |
| sudo snap install notepadqq |

### 温度监控

先安装必须项：

|  |
| --- |
| sudo apt install lm-sensors hddtemp  sudo sensors-detect |

用命令监控：

|  |
| --- |
| watch -n 2 sensors #监控CPU温度  watch -n 2 nvidia-smi #监控GPU温度 |

图形化监控：

|  |
| --- |
| sudo apt install psensor |

### 英伟达显卡驱动

|  |
| --- |
| ubuntu-drivers devices #显示显卡信息  sudo ubuntu-drivers autoinstall #自动安装推荐的版本  sudo apt install <driver\_name\_in\_the\_list> #安装指定的版本，e.g. sudo apt install nvidia-340 |

详细教程参看 [Ubuntu 18.04 安装 NVIDIA 显卡驱动](https://zhuanlan.zhihu.com/p/59618999)

### indicator-sound-switcher / Sound Switcher Indicator

**简介：**音频切换器

**安装：**

|  |
| --- |
| sudo apt-add-repository ppa:yktooo/ppa  sudo apt-get update  sudo apt-get install indicator-sound-switcher |

**官网：**[Sound Switcher Indicator](https://yktoo.com/en/software/sound-switcher-indicator/)

**GitHub网址：**[indicator-sound-switcher](https://github.com/yktoo/indicator-sound-switcher)

### PulseAudio Volume Control

**简介：**音频切换器

[PulseAudio Volume Control](https://freedesktop.org/software/pulseaudio/pavucontrol/#download)

### 桌面

1、安装以下组件

|  |
| --- |
| sudo apt install gnome-tweak-tool gnome-shell-extensions chrome-gnome-shell |

2、用火狐浏览器（不支持chrome）打开[GNOME Shell Extensions](https://link.zhihu.com/?target=https%3A//extensions.gnome.org/)安装所需的插件，以下是几个比较好的插件：

* [User theme](https://extensions.gnome.org/extension/19/user-themes/)：使shell主题可以使用桌面主题，shell即为顶部栏，shell主题和桌面主题不一样，是个单独的模块。
* [Dash to dock](https://extensions.gnome.org/extension/307/dash-to-dock/)：更改dock的样式，可以使图标居中
* [Hide top bar](https://extensions.gnome.org/extension/545/hide-top-bar/)：可以设置自动隐藏顶部栏
* [Dash to Panel](https://extensions.gnome.org/extension/1160/dash-to-panel/)和[Dynamic panel transparency](https://extensions.gnome.org/extension/1011/dynamic-panel-transparency/)：类似Windows的任务栏，后一个设置透明效果

### 主题

#### MacOS

* [主题](https://www.opendesktop.org/s/Gnome/p/1241688)
* [图标](https://www.opendesktop.org/s/Gnome/p/1102582/)
* [shell](https://www.opendesktop.org/s/Gnome/p/1013741/)：需要安装 [User theme](https://extensions.gnome.org/extension/19/user-themes/) 插件才能生效

### 安装字体

方法1：系统级

|  |
| --- |
| sudo mkdir usr/share/fonts/<dir\_name>  sudo cp <xxx.ttf> /usr/share/fonts/<dir\_name>  sudo chmod 644 /usr/share/fonts/<dir\_name>/\*  cd /usr/share/fonts/<dir\_name>  sudo mkfontscale  sudo mkfontdir |

方法2：用户级

在*home/<user\_name>*下建立*.fonts*的隐藏文件夹，将字体文件拷贝到这个文件夹下即可，只对当前用户生效。该方法等同于*字体管理器*软件的底层实现方法。

系统字体选择

终端等宽字体

* Monospace：系统自带，推荐
* Inconsolata：*sudo apt-get install fonts-inconsolata*，推荐

## CentOS专区

### 离线状态下VMware安装CentOS及配置

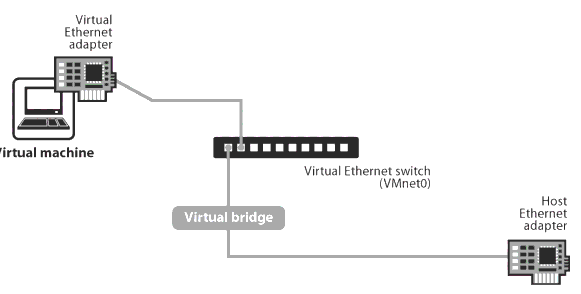
**1) 安装**

官网下载**CentOS 7 Everything**，安装的时候选择**minimal**安装即可。

**2) Xshell连接**

① 桥接模式

桥接网络是指本地物理网卡和虚拟网卡通过VMnet0虚拟交换机进行桥接，物理网卡和虚拟网卡在拓扑图上处于同等地位，那么物理网卡和虚拟网卡就相当于处于同一个网段，虚拟交换机就相当于一台现实网络中的交换机，所以两个网卡的IP地址也要设置为同一网段。桥接模式如下图所示：

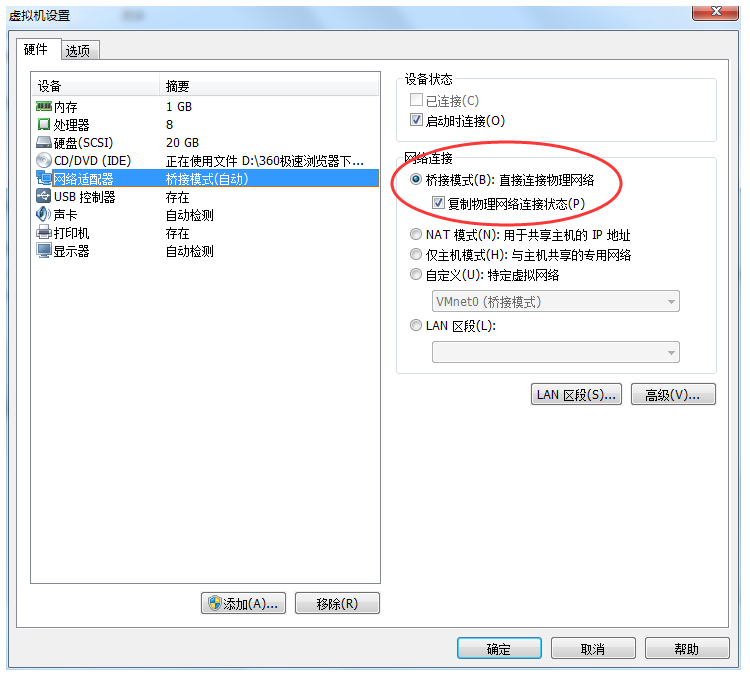


② 配置环境

VMWare Workstation 12 + CentOS + Xshell

③ 配置步骤

1、在VMWare主菜单**WorkStation > 虚拟机 > 设置**，将网络连接方式设置为**桥接模式**，并在**复制物理网络连接状态**前打钩。这里是指将创建一个虚拟的网卡，而该虚拟网卡所有的状态均是从实际的物理网卡中复制过来的。



2、在VMWare主菜单**WorkStation > 编辑 > 虚拟网络编辑器 > 选择VMnet0 > 将其桥接到“Realtek PCIe GBE Family Controller”**。这里**Realtek PCIe GBE Family Controller**是真实的网卡，根据实际情况而定，而VMnet0是虚拟交换机。

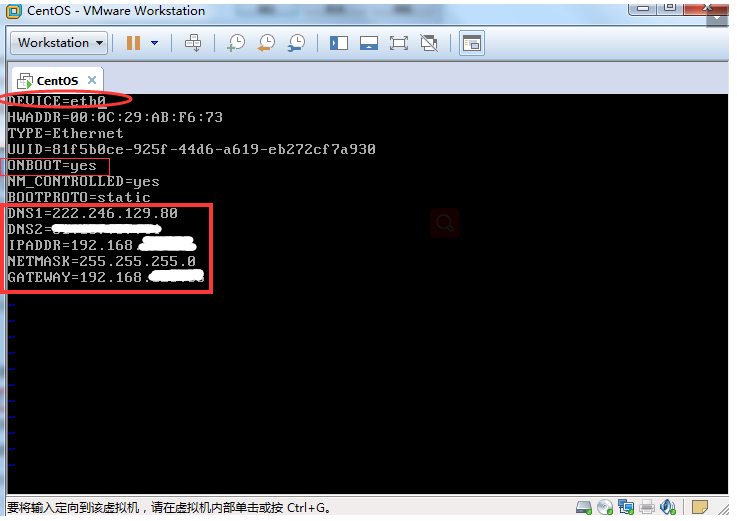
3、在VMWare下打开CentOS后，输入如下语句，进行相关的网络配置：

|  |
| --- |
| cd /etc/sysconfig/network-scripts #进入该目录查看网卡信息，第一个就是本地的默认网卡，修改它，例如这里是：ifcfg-eth0  vim ifcfg-eth0 |

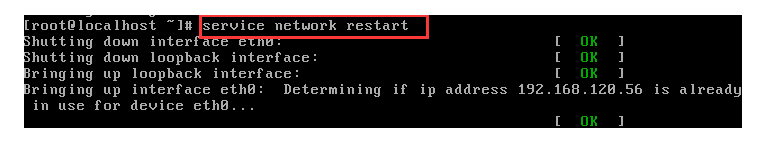
4、打开**ifcfg-eth0**文件，设置如下：

|  |
| --- |
| DEVICE=eth0 //虚拟网卡名  ONBOOT=yes //指明在系统启动时激活网卡  BOOTPROTO=static //这里 dhcp 一定要改为 static，意思为静态 IP 否则配置不会生效  DNS1=xxx.xxx.xxx.xxx //主用 DNS 地址  DNS2=xxx.xxx.xxx.xxx //备用 DNS 地址  IPADDR=xxx.xxx.xxx.xxx //虚拟机 IP 地址（注意：这里的虚拟 IP 地址必须与主机 IP 地址在同一网段内）  NETMASK=xxx.xxx.xxx.xxx //子网掩码  GATEWAY=xxx.xxx.xxx.xxx //网关 IP（与真实网关地址一致） |

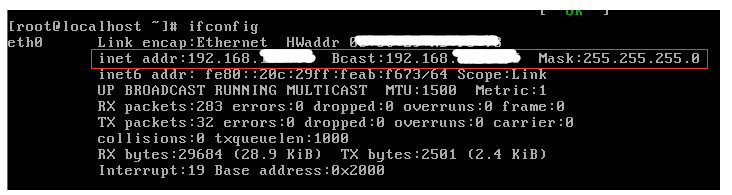
后边几个配置项，打开主机的物理网卡相关信息，并依据填写。



5、完成以上配置后，退出并保存。并输入*service network restart*，重新启动网络使配置生效。



6、输入*ifconfig*，如配置成功，会出现刚才配置的IP等信息，如下图所示。



7、Linux和主机相ping，如果能ping通，说明虚拟机网络配置已成功。

8、有些虚拟系统可能没有启动ssh服务，因此还要输入*servcie sshd restart*。

9、进入Xshell并连接，ssh端口号默认为22。

**3) yum本地镜像源设置**

1、准备CentOS ISO文件

方法1：**把CentOS的镜像放在本地PC硬盘上 > 打开VMware Workstation > 虚拟机 > 设置 > 硬件 > CD/DVD(IDE) > 使用ISO映像文件 > 选择PC上的CentOS镜像 > 设备状态选择：启动时连接**

方法2：这里也可以把CentOS的映像通过WinSCP上传到Linux系统上，例如上传到*/usr/local/src*目录中

2、挂载镜像文件

|  |
| --- |
| mkdir /mnt/cdrom # 新建光盘目录  # 如果第 1 步用方法 1，这样挂载：  mount /dev/cdrom /mnt/cdrom  # 如果第 1 步用方法 2，这样挂载：  mount -t iso9660 -o loop /usr/local/src/CentOS-7.0-1406-x86\_64-Everything.iso /mnt/cdrom/ |

3、设置开机自动挂载系统镜像文件：

*vim /etc/fstab*在末尾添加*/dev/cdrom /mnt/cdrom/ iso9660 defaults,ro,loop 0 0*

*:wq* #保存并退出

**注：**iso9660 CD-ROM光盘的标准文件系统

4、配置本地yum源文件

先备份并删除*/etc/yum.repos.d*所有的*.repo*文件

*vim /etc/yum.repos.d/centos7-iso.repo*

|  |
| --- |
| [centos7-iso] # 自定义  name=centos7-iso # 自定义  baseurl=file:///mnt/cdrom # 本地镜像文件路径  enabled=1 # 1为启动yum源，0为禁用  gpgcheck=1 # 1为检查 GPG-KEY，0为不检查（如果后续使用 yum 安装软件提示 GPG-KEY 相关问题，设为 0 即可）  gpgkey=file:///media/cdrom/RPM-GPG-KEY-CentOS-7 # GPG-KEY文件路径 |

**安装：**

|  |
| --- |
| yum --disablerepo=\\* --enablerepo=centos7-media clean all # 清除缓存  yum --disablerepo=\\* --enablerepo=centos7-media makecache # 缓存本地yum源包信息  yum --disablerepo=\\* --enablerepo=centos7-media install tigervnc # 使用镜像源安装软件 |

**4) 设置共享文件夹**

1、在**虚拟机 > 设置 > 选项 > 共享文件夹**中设置好需要共享的文件夹

2、安装**open-vm-tools**，系统自带，已经安装好了，不用再安装了

3、输入如下命令挂载主机文件

|  |
| --- |
| mkdir /mnt/hgfs  vmhgfs-fuse .host:/ /mnt/hgfs # 挂载所有的共享文件夹  vmhgfs-fuse .host:/foo/bar /mnt/hgfs # 只挂载/foo/bar文件夹 |

**5) 修改主机名**

1、设置主机名

|  |
| --- |
| [root@localhost ~]# hostnamectl set-hostname zdylinux  [root@localhost ~]# hostnamectl --pretty  [root@localhost ~]# hostnamectl --static  zdylinux  [root@localhost ~]# hostnamectl --transient  zdylinux |

2、手动更新*/etc/hosts*

|  |
| --- |
| vim /etc/hosts  # 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  127.0.0.1 zdylinux  # ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  ::1 zdylinux |

3、重启*reboot -f*

## Manjaro专区

### 中文输入法

1**、**安装安装输入法模块

|  |
| --- |
| sudo pacman -S fcitx-im #按⌈Enter⌋安装全部输入法模块 |

2**、**安装输入法配置工具

|  |
| --- |
| sudo pacman -S fcitx-configtool |

3**、**在*/etc/pacman.conf*配置镜像源地址

4、安装某一输入法，也可不安装，自带的拼音就是fcitx-libpinyin，这个是比较好用的。可参看[Fcitx(简体中文)-Archlinux Wiki](https://wiki.archlinux.org/index.php/Fcitx_(%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)查看fcitx介绍，以及支持的其它输入法

5**、**修改HOME目录下*.xprofile*文件，没有需要创建

|  |
| --- |
| vim ~/.xprofile |

添加以下内容并保存：

|  |
| --- |
| export GTK\_IM\_MODULE=fcitx  export QT\_IM\_MODULE=fcitx  export XMODIFIERS=@im=fcitx |

*~/.xprofile*：用户级配置文件，用户每次以**X Windows**登陆方式登陆，则执行该配置文件

*~/.profile*：用户级配置文件，用户每次以**命令行**登陆方式登陆，则执行该配置文件

6**、**可能需要重启后生效

|  |
| --- |
| reboot |

7**、**输入特殊字符的两种方法参看：搜狗输入法

### google chrome

|  |
| --- |
| sudo pacman -S google-chrome |

### WPS Office以及WPS Office字体

|  |
| --- |
| sudo pacman -S wps-office #WPS，中科大源  sudo pacman -S ttf-wps-fonts #WPS 字体，中科大源 |

### aurman

**简介：**AUR仓库软件下载工具

参看[AUR helpers(简体中文)](https://wiki.archlinux.org/index.php/AUR_helpers_(%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)#%E5%9B%BE%E5%BD%A2%E5%8C%96%E5%B7%A5%E5%85%B7)查看更多下载工具

参看[Arch User Repository (简体中文)](https://wiki.archlinux.org/index.php/Arch_User_Repository_(%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87))查看AUR说明，AUR软件安装方法

**安装步骤：**

|  |
| --- |
| git clone https://github.com/polygamma/aurman  cd aurman  makepkg -si #会提示未知的公共密匙 xxxx 错误 ...  gpg --receive-keys <未知的公共密匙 xxxx>  makepkg -si #再次运行，完成安装 |

### indicator-sound-switcher

**简介：**音频切换器，可以切换音频输出源，尤其是电脑连接电视时，可以选择使用电视的音频源。

|  |
| --- |
| aurman -S indicator-sound-switcher |

### QQ

|  |
| --- |
| pacman -S deepin.com.qq.office |

### TIM

|  |
| --- |
| aurman -S deepin-wine-tim #AUR有很多其它版本 |

### 微信

|  |
| --- |
| aurman -S deepin-wine-wechat #AUR有很多其它版本 |

### shadowsocks

**简介：**代理软件

|  |
| --- |
| sudo pacman -S shadowsocks-qt5 |

### electron-ssr

**简介：**代理软件

|  |
| --- |
| sudo pacman -S electron-ssr |

### Typora

**简介：**一款好用的 Markdown编辑器

|  |
| --- |
| sudo pacman -S typora |

### 百度网盘

|  |
| --- |
| sudo pacman -S baidunetdisk |

### Visual Studio Code

|  |
| --- |
| sudo pacman -S visual-studio-code-bin |

### gedit

**简介：**图形化文本编辑工具，对比vim和nano

|  |
| --- |
| sudo pacman -S gedit |

### nload

**简介：**网速监控

|  |
| --- |
| sudo pacman -S nload |

### gucharmap

**简介：**GNOME 的字符映射表

|  |
| --- |
| sudo pacman -S gucharmap |

### OneDrive

**项目GitHub主页：**[electron-ssr](https://github.com/skilion/onedrive/tree/master)

**安装步骤：**

1**、**安装依赖项

|  |
| --- |
| sudo pacman -S curl sqlite dlang |

2**、**安装OneDrive

|  |
| --- |
| git clone https://github.com/skilion/onedrive.git  cd onedrive  make  sudo make install |

3、其它操作参看：OneDrive

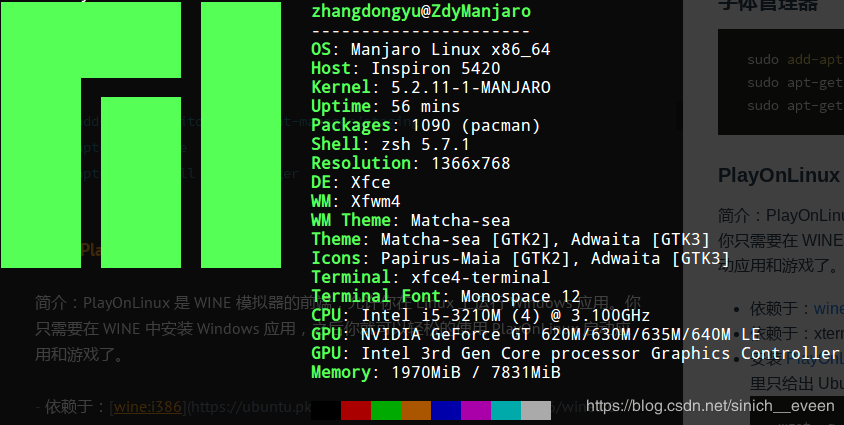
### neofetch

**简介：**显示系统信息

**安装：**

|  |
| --- |
| sudo pacman -S neofetch |

**效果图：**



### 简单天气指示器

**简介：**简单天气指示器是用Python开发的开源天气提示应用。它自动侦查你的位置，并显示你天气信息像温度，下雨的可能性，湿度，风速和可见度。

**GitHub网址：**[UbuntuIndicatorWeather](https://github.com/kasramp/UbuntuIndicatorWeather)

**安装方法：**

|  |
| --- |
| aurman -S ubuntu-indicator-weather |

### notepadqq

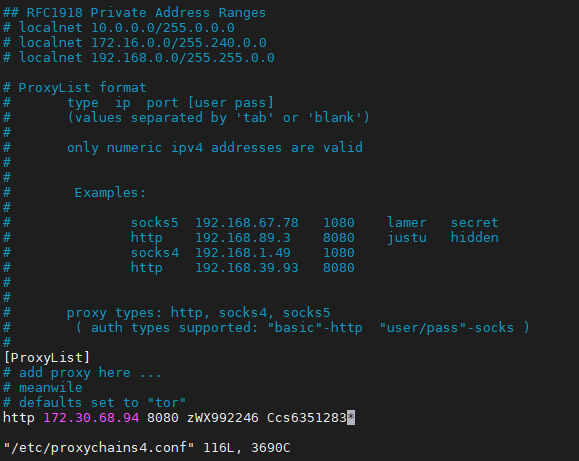
**简介：**Linux版的notepad++

|  |
| --- |
| sudo pacman -S notepadqq |

## 通用专区

### Trojan：sock5代理软件

参看：5、上边设置的是sock5代理，这里提供一个设置http代理的实例（鼎桥公司）：



6、注意：*proxy\_dns*有时候需要注释（例如，自己电脑设置sock5代理），有时候又不能注释（例如，鼎桥http代理），具体看情况，那种情况OK用那种。

安装Trojan实现系统代理

### proxychains：让终端命令使用socks5/http代理

参看：安装proxychains，让终端命令使用socks5/http代理

### sqlite3

进入[官网](http://www.sqlite.org)，下载最新安装包。

这里以写文章时的最新包*sqlite-autoconf-3070603.tar.gz*为例。

**安装：**

|  |
| --- |
| tar xvfz sqlite-autoconf-3070603.tar.gz  cd sqlite-autoconf-3070603  ./configure #可能需要执行两次才会生成 Makefile  make -j4 #-j4 代表使用四线程编译，如果 CPU 支持的线程更多，这里可以设置对应的线程数  make install |

**安装完成后文件存储路径：**

* 可执行文件：*/usr/local/bin*
* 头文件：*/usr/local/include*
* 动态库：*/usr/local/lib*

### ShellCheck：脚本语法检查工具

**教程参考来源：**[ShellCheck – A Tool That Shows Warnings and Suggestions for Shell Scripts](https://www.tecmint.com/shellcheck-shell-script-code-analyzer-for-linux/)

**简介：**

ShellCheck是一个静态分析工具，它显示有关bash脚本中错误代码的警告和建议。

**ShellCheck主要做三件事：**

* 它指出并解释了典型的初学者的语法问题，这些问题会导致shell发出错误的错误消息。
* 它指出并解释了导致shell表现异常和违反直觉的典型中级语义问题。
* 它还指出了一些细微的警告，极端情况和陷阱，它们可能会导致高级用户的其他工作脚本在将来的情况下失败。

**1) 在Web端使用**

<https://www.shellcheck.net/>

**2) 在shell终端使用**

1、安装ShellCheck软件，见 **(3) 集成vim使用**

2、编写脚本，例如*test.sh*，然后在shell终端中运行如下命令：

|  |
| --- |
| shellcheck test.sh |

之后会在shell终端显示语法错误信息。

**3) 集成vim使用**

1、安装ShellCheck

On Debian/Ubuntu

|  |
| --- |
| apt-get install shellcheck |

On RHEL/CentOS

|  |
| --- |
| yum -y install epel-release  yum install ShellCheck |

On Fedora

|  |
| --- |
| dnf install ShellCheck |

2、安装Pathogen

**在线安装：**

|  |
| --- |
| mkdir -p ~/.vim/autoload ~/.vim/bundle #创建目录  curl -LSso ~/.vim/autoload/pathogen.vim https://tpo.pe/pathogen.vim #下载 pathogen.vim 放入到 ~/.vim/autoload/ 下 |

**离线安装：**

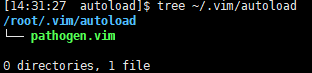
<https://tpo.pe/pathogen.vim>指向的其实是一个代码文本文件，如果因为网络问题导致无法下载可以在线复制https://tpo.pe/pathogen.vim指向的内容放入一个文本文档中，把该文本文档命名为*pathogen.vim*，然后把*pathogen.vim*放入到*~/.vim/autoload/*目录下，**并用vim打开，把dos换行符替换为linux换行符**。

这里提供已经复制好的*pathogen.vim*，且换行符已经替换为linux换行符：

附件：pathogen.vim

下载附件后把*pathogen.vim*放入*~/.vim/autoload/*目录下。

**文件结构图：**



3、添加如下内容到*~/.vimrc*文件中（没有的话需要自己创建）

|  |
| --- |
| execute pathogen#infect() |

4、安装syntastic

**在线安装：**

|  |
| --- |
| cd ~/.vim/bundle  git clone --depth=1 <https://github.com/vim-syntastic/syntastic.git> |

**离线安装：**

如果因为网络原因无法下载需要手动去网页下载，并解压复制到*~/.vim/bundle*目录中。

这里提供已经下载好的：

附件：syntastic.zip

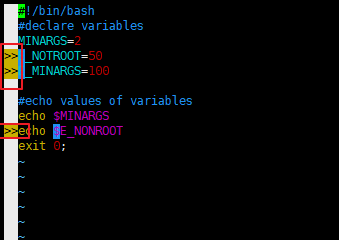
下载后需要把*syntastic.zip*解压，并把解压后得到的文件夹*syntastic*放到*~/.vim/bundle*目录中。

**文件结构如图：**

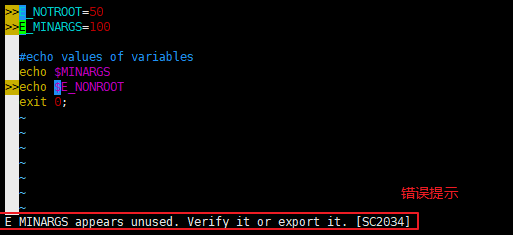


5、开始使用

关闭vim软件并重新打开，编写脚本，当保存脚本(*:w*)的时候如果脚本有语法错误，会出现如下画面：



错误行前边会有“>>”标识，光标移动到该行，最下边会出现提示：



若不出现提示，可能需要在vim中执行以下命令：

|  |
| --- |
| :Helptags |

**4) 附件**

32-F-ShellCheck

### pandoc：文档格式转换工具

**1) 网址**

**官网：**<https://pandoc.org/>

**GitHub网址：**<https://github.com/jgm/pandoc>

**2) 安装**

以Ubuntu为例：

1、进入网址<https://github.com/jgm/pandoc/releases/tag/2.7.3>下载**pandoc-2.7.3-1-amd64.deb**

2、*sudo dpkg -i <package\_name>*

**3) 使用**

① markdown转pdf

1、安装转pdf所必需的包：*sudo apt install texlive-xetex*，如果提示下载错误先*sudo apt update*之后，再运行*sudo apt install texlive-xetex*。安装下载过程比较长，中间会弹出一个界面询问东西，直接点击两次**Enter**即可。

2、创建转pdf所需要的LaTeX模板

|  |
| --- |
| mkdir -p ~/.pandoc/templates  cd ~/.pandoc/templates  vim pdf.template #名字随意 |

3、将以下内容添加到*pdf.template*文件中

**版本1：有标题**

附件：v1.zip

**版本2：无标题**

附件：v2.zip

如果没有所需字体，参看：安装字体

4、转换命令

a. 单个文件转换

|  |
| --- |
| pandoc <xxx.md> --pdf-engine=xelatex --template=pdf.template -o <xxx.pdf> |

b. 多文件逐个转换

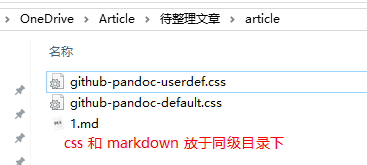
附件：z-pandoc-to-pdf.sh

脚本功能：读取一个目录下指定类型的文件，然后通过管道和*xargs*逐个转换成pdf

② markdown转html（应用css样式）

1、准备css样式

把css样式放在和需要转换的markdown同路径下，不能放在*/root/.pandoc/*下，经试验这样识别不了。



css样式1：（github markdown样式）

附件：github-pandoc-default.css

css样式2：（在“css样式1”的基础上修改）

附件：github-pandoc-default.css

2、转换命令

a. 单个文件转换

|  |
| --- |
| pandoc --toc -s -c github-pandoc-default.css --metadata title="<title\_name>" <xxx.md> -o <xxx.html> |

b. 多文件逐个转换

附件：z-pandoc-to-html.sh

脚本功能：读取一个目录下指定类型的文件，然后通过管道和*xargs*逐个转换成html

3) 附件

32-F-pandoc

# 系统配置

## Ubuntu专区

### Ubuntu 18.04 LTS更改软件源为阿里云源

1、备份*/etc/apt/sources.list*以防万一

2、删除*/etc/apt/sources.list*原有内容，并改为以下内容：

|  |
| --- |
| # 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释  deb https://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse  # deb-src https://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse  deb https://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse  # deb-src https://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse  deb https://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse  # deb-src https://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse  deb https://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse  # deb-src https://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse  # 预发布软件源，不建议启用  # deb https://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse  # deb-src https://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse |

3、更新源

|  |
| --- |
| sudo apt update |

**注意：**

* 这是阿里云源，也可选用其它的
* 这是18.04 LTS的，可以更改版本代号应用于其它版本

### 中文字体渲染美化+去模糊

**步骤：**

1、解压安装**lulinux\_fontsConf\_181226.tar.gz**，按里面的安装说明操作

2、开启字体渲染

打开**unity-tweak-tool**，更改字体：



3、附件

32-F-Ubuntu中文字体渲染+去模糊

### 设置代理

#### 设置HTTP、HTTPS、FTP、SOCK5代理

**1) 只在当前shell生效**

|  |
| --- |
| export http\_proxy="http://<user>:<password>@<proxy\_server>:<port>"  export https\_proxy="http://<user>:<password>@<proxy\_server>:<port>"  export http\_proxy="socks5://127.0.0.1:1080"  export https\_proxy="socks5://127.0.0.1:1080"  export ftp\_proxy=http://<user>:<password>@<proxy\_server>:<port> |

**或**

|  |
| --- |
| export all\_proxy=http://<user>:<password>@<proxy\_server>:<port>  export all\_proxy="socks5://127.0.0.1:1080" |

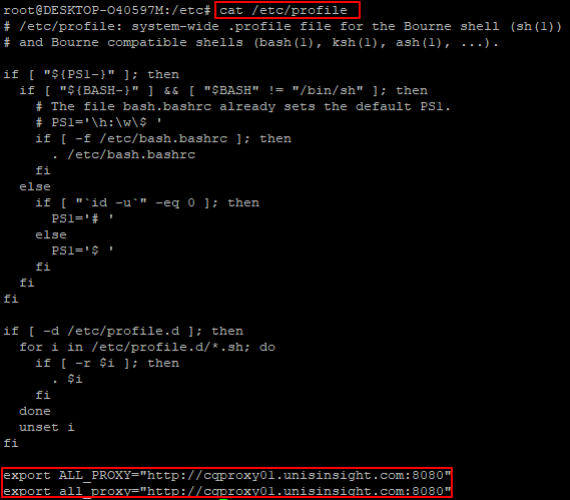
**注意：**

* 其中*<user>:<password>*不一定需要，如果没有可以不设置，则变为*export http\_proxy="http://<proxy\_server>:<port>"*
* 有些软件识别不了小写版本，可以在再设置一份大写版本*HTTP\_PROXY HTTPS\_PROXY FTP\_PROXY ALL\_PROXY*

**2) 全局生效**

|  |
| --- |
| vim /etc/profile #编辑全局配置文件  #TODO -> 在文件末尾添加 1.1.1 中的代理代码，保存后关闭文件  source /etc/profile #使配置文件生效 |

**配置文件实例：**



**3) http\_proxy、https\_proxy、ftp\_proxy、all\_proxy、no\_proxy更像是一种约定俗成的说法，而不是标准，关于它们的讨论参看：**

* [Are HTTP\_PROXY, HTTPS\_PROXY and NO\_PROXY environment variables standard?](https://superuser.com/questions/944958/are-http-proxy-https-proxy-and-no-proxy-environment-variables-standard)
* [HTTP\_PROXY,HTTPS\_PROXY,NO\_PROXY,ALL\_PROXY来自什么协议？](http://blog.champbay.com/2019/06/06/http_proxyhttps_proxyno_proxyall_proxy%E6%9D%A5%E8%87%AA%E4%BB%80%E4%B9%88%E5%8D%8F%E8%AE%AE%EF%BC%9F/)

#### 安装proxychains，让终端命令使用socks5/http代理

**教程参考来源：**[Ubuntu安装proxychains，让终端命令使用socks5代理…](https://blog.popkx.com/Ubuntu-install-proxychains-let-terminal-using-socks5-proxy-speed-up-downloading/)

**注意：**本教程是proxychains 3的，事实上写的时候已经有proxychains 4了，完全可以参照此方法安装proxychains 4。

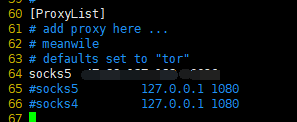
1、安装proxychains

|  |
| --- |
| sudo apt install tor  sudo apt install proxychains |

完成后便可使用proxychains程序。

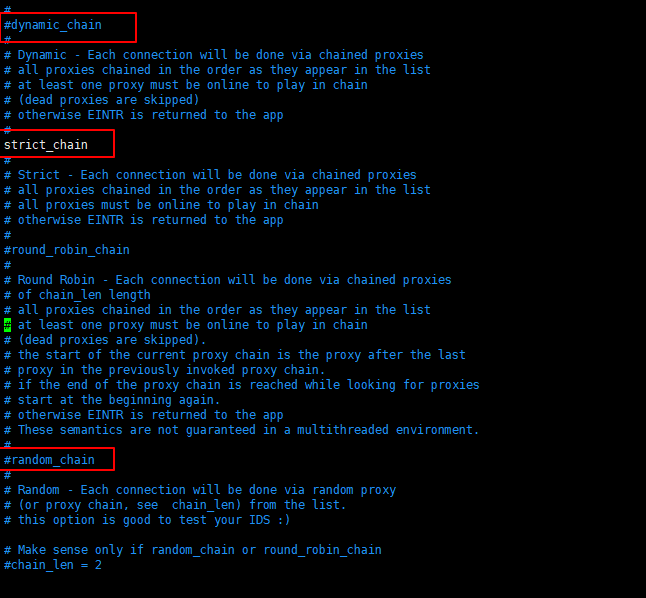
2、使用proxychains

主要需要配置*/etc/proxychains.conf*文件，打开之，在*[ProxyList]*项添加socks5服务的ip:port，如下图所示：



正如proxychains的名字所示，它是一个“代理链”，因此如果填写了多个代理，那么proxychains将会将这些代理组成链。

里边有三种模式，英文解释说的也比较清楚，默认是*strict\_chain*模式：



3、若在稍后的使用中出现类似于下面这样的错误：

|  |
| --- |
| $ proxychains curl baidu.com  ProxyChains-3.1 (http://proxychains.sf.net)  |DNS-request| baidu.com  |S-chain|-<>-127.0.0.1:1080-<--timeout  |DNS-response|: baidu.com does not exist  curl: (6) Could not resolve host: baidu.com |

则，取消*proxy\_dns*，如下图所示：

取消 proxy_dns

配置完毕后，执行下面的命令尝试使用proxychains：

|  |
| --- |
| $ proxychains ping baidu.com  ProxyChains-3.1 (http://proxychains.sf.net)  ERROR: ld.so: object 'libproxychains.so.3' from LD\_PRELOAD cannot be preloaded (cannot open shared object file): ignored. |

根据错误信息，是没有从*LD\_PRELOAD*里找到*'libproxychains.so.3'*库。解决这个问题也很简单，先使用*which*命令查看*proxychains*所在目录：

|  |
| --- |
| $ which proxychains  /usr/bin/proxychains |

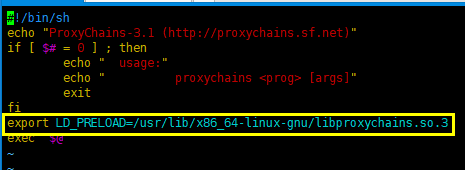
然后在*/usr*目录搜索*'libproxychains.so.3'*：

|  |
| --- |
| $ find /usr -name "libproxychains.so.3"  /usr/lib/x86\_64-linux-gnu/libproxychains.so.3 |

接着便可打开proxychains文件：

|  |
| --- |
| $ sudo vim /usr/bin/proxychains |

将上面搜索到的库路径填入：



4、成功使用

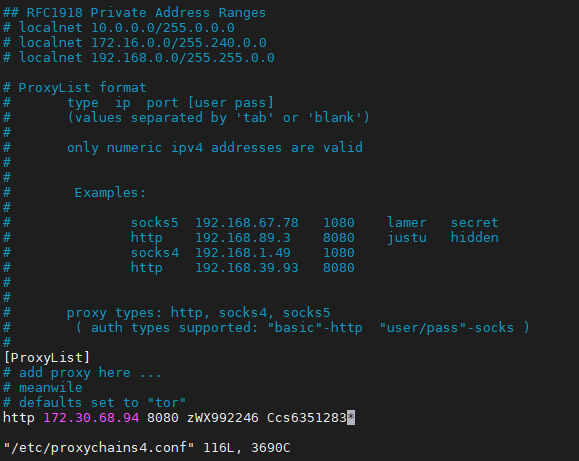
经过上面的配置和修改，就可以正常使用proxychains程序进行Ubuntu的终端命令socks5代理了，方法很简单，只需要在命令前加上proxychains即可，例如：

|  |
| --- |
| $ proxychains curl baidu.com  $ sudo proxychains pip install numpy  $ sudo proxychains apt install gcc |

等。看到类似于下面这样的输出，即可表明proxychains正常连接到socks5服务了：

|  |
| --- |
| ...  |S-chain|-<>-<server ip:port>-<><>-151.101.40.223:443-<><>-OK  |S-chain|-<>-<server ip:port>-<><>-151.101.0.223:443-<><>-OK  |S-chain|-<>-<server ip:port>-<><>-151.101.25.63:443-<><>-OK  ... |

5、上边设置的是sock5代理，这里提供一个设置http代理的实例（鼎桥公司）：



6、注意：*proxy\_dns*有时候需要注释（例如，自己电脑设置sock5代理），有时候又不能注释（例如，鼎桥http代理），具体看情况，那种情况OK用那种。

#### 安装Trojan实现系统代理

本教程以购买的这个服务为例：<https://shadowsocks.com/>

教程来源网址：<https://portal.shadowsocks.nz/knowledgebase/160/>

Trojan教程参看：<https://tlanyan.me/trojan-tutorial/#system>

1、下载客户端

访问<https://dl.trojan-cdn.com/trojan/linux/>下载

访问[Github](https://github.com/trojan-gfw/trojan/releases)下载

下载*trojan-[版本号]-linux-amd64.tar.xz*文件

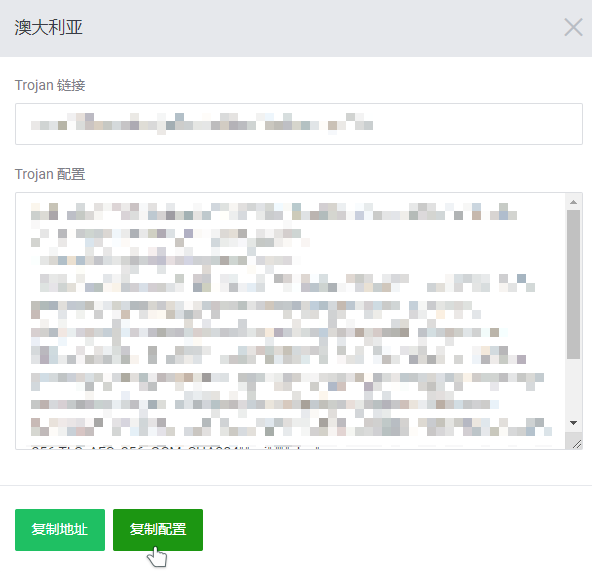
2、获取配置

命令行客户端只能设置单节点。

登入到客户中心，依次访问**产品服务 > 我的产品与服务**，查看Trojan服务对应的**云加速服务-Lite / Pro**服务器信息。

在节点信息最后一列：

点击齿轮图标打开单节点的配置窗口，点击复制配置即可复制节点配置。



3、配置客户端

解压客户端后，进入客户端的目录，使用文本编辑器编辑*config.json*文件，使用**2、获取配置**中复制的配置替换全部的内容后保存。

然后使用下面的命令运行客户端即可

|  |
| --- |
| sudo ./trojan |

如果希望后台运行，请执行（当无法使用时，请不要后台运行，以便查看运行日志）

|  |
| --- |
| sudo ./trojan & |

退出的话，请运行下面的命令。

|  |
| --- |
| pkill -f trojan |

4、设置代理

不同于图形客户端，命令行客户端运行后不会对系统或其它软件有任何影响，需要手动设置系统代理或是在浏览器内安装扩展使用。这里启动后可以配合proxychain使用。

#### 为WSL Ubuntu 18.04 LTS配置Windows系统正在使用的sock5代理

**参考来源：**<https://segmentfault.com/a/1190000015913747>

1、安装**python pip**（如果安装失败，关闭终端然后重新打开一个终端，重新安装）

|  |
| --- |
| apt install python-pip |

2、升级**pip**（如果安装失败，关闭终端然后重新打开一个终端，重新安装）

|  |
| --- |
| pip install --upgrade pip |

3、安装**genpac**工具（如果安装失败，关闭终端然后重新打开一个终端，重新安装）

|  |
| --- |
| pip install genpac |

4、生成配置

|  |
| --- |
| #TODO -> 切换到想存配置文件的路径  genpac --proxy="SOCKS5 127.0.0.1:1080" -o autoproxy.pac --gfwlist-url="https://raw .githubusercontent.com/gfwlist/gfwlist/master/gfwlist.txt" |

5、编辑*/etc/profile*在文件最后添加，具体端口配置和ss客户端保持一致。这里可以参考2.1.1.1的两种设置方法。

|  |
| --- |
| export http\_proxy=http://127.0.0.1:1080  export https\_proxy=http://127.0.0.1:1080  export ftp\_proxy=http://127.0.0.1:1080 |

6、*source /etc/profile* 让配置生效

7、使用*curl www .google.com*查看代理是否开启成功。

#### 分别给软件设置代理

##### (1) 为apt设 http代理

**方法1：**

1、Create a new configuration file named *proxy.conf*

|  |
| --- |
| sudo touch /etc/apt/apt.conf.d/proxy.conf |

2、Open the *proxy.conf* file in a text editor

|  |
| --- |
| sudo vim /etc/apt/apt.conf.d/proxy.conf |

3、Add the following line to set your HTTP proxy

|  |
| --- |
| Acquire::http::Proxy "http://<user>:<password>@<proxy\_server>:<port>"; |

4、Add the following line to set your HTTPS proxy

|  |
| --- |
| Acquire::https::Proxy "http://<user>:<password>@<proxy\_server>:<port>"; |

5、Save your changes and exit the text editor

6、Your proxy settings will be applied the next time you run apt.

**方法2：**

Create a new file named *proxy.conf* under the */etc/apt/apt.conf.d* directory, and then add the following lines:

|  |
| --- |
| Acquire {  HTTP::proxy "http://<user>:<password>@<proxy\_server>:<port>";  HTTPS::proxy "http://<user>:<password>@<proxy\_server>:<port>";  } |

### WSL Ubuntu 18.04 LTS系统路径

|  |
| --- |
| C:\Users\<user>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows\_79rhkp1fndgsc\LocalState\rootfs |

### 将Ubuntu默认的中文文件夹名称改英文

**方法1：**

**打开系统设置 > 语言支持 > 将english拖动到最上端 > 重启系统。**

重启后，会提示更新文件名称，更新后再将语言「中文」拖动到顶部，重启系统。

**方法2：**

编辑*~/.config/user-dirs.dirs*文件

|  |
| --- |
| vim ~/.config/user-dirs.dirs |

修改文件内容为：

|  |
| --- |
| XDG\_DESKTOP\_DIR="$HOME/Desktop"  XDG\_DOWNLOAD\_DIR="$HOME/Downloads"  XDG\_TEMPLATES\_DIR="$HOME/Templates"  XDG\_PUBLICSHARE\_DIR="$HOME/Public"  XDG\_DOCUMENTS\_DIR="$HOME/Documents"  XDG\_MUSIC\_DIR="$HOME/Music"  XDG\_PICTURES\_DIR="$HOME/Pictures"  XDG\_VIDEOS\_DIR="$HOME/Videos" |

再在「文件」应用里边把现有的中文名字改为对应的英文名字。

**方法3：**

打开终端，在终端中输入命令：

|  |
| --- |
| export LANG=en\_US  xdg-user-dirs-gtk-update |

跳出对话框询问是否将目录转化为英文路径，同意并关闭。

再次在终端中输入命令：

|  |
| --- |
| export LANG=zh\_CN  xdg-user-dirs-gtk-update |

重新启动系统，系统会提示更新文件名称，选择不再提示，并取消修改。

### 安装 deb 缺少依赖项

|  |
| --- |
| sudo dpkg -i <package\_name.deb> #执行安装命令，提示缺少依赖，没成功安装  sudo apt-get -f -y install #安装所需依赖  sudo dpkg -i <package\_name.deb> #再次执行，即可安装 |

### 安装preload来加速应用载入时间

**简介：**Preload 是一个后台运行的守护进程，它分析用户行为和频繁运行的应用。

|  |
| --- |
| sudo apt-get install preload |

### 减少过热

以下两个工具可以用来减少过热，使 Ubuntu 获得更好的系统表现，即 TLP 和 CPUFREQ。

**TLP**

|  |
| --- |
| sudo add-apt-repository ppa:linrunner/tlp  sudo apt-get update  sudo apt-get install tlp tlp-rdw  sudo tlp start |

**CPUFREQ**

|  |
| --- |
| sudo apt-get install indicator-cpufreq |

重启电脑并使用 Powersave 模式

**打开英伟达显卡设置程序 > PRIME Profiles > Inter（Power Saving Mode）**，选择使用集成显卡。

### 开启命令自动补全

1、安装**bash-completion**（一般都安装了）

|  |
| --- |
| apt install bash-completion |

2、利用vim编辑器打开*/etc/bash.bashrc*文件（需要root权限）

|  |
| --- |
| sudo vim /etc/bash.bashrc |

3、找到文件中的下列代码：

|  |
| --- |
| #enable bash completion in interactive shells  #if ! shopt -oq posix; then  # if [-f /usr/share/bash-completion/bash\_completion ]; then  # . /usr/share/bash-completion/bash\_completion  # elif [ -f /etc/bash\_completion]; then  # . /etc/bash\_completion  # fi  #fi |

4、将注释符号*#*去掉，即改成：

|  |
| --- |
| #enable bash completion in interactive shells  if ! shopt -oq posix; then  if [-f /usr/share/bash-completion/bash\_completion ]; then  . /usr/share/bash-completion/bash\_completion  elif [ -f /etc/bash\_completion]; then  . /etc/bash\_completion  fi  fi |

5、最后*source*一下*/etc/bash.bashrc*即可，即：

|  |
| --- |
| sudo source /etc/bash.bashrc |

## CentOS专区

### CentOS8更改软件源为阿里云源

1、备份现有源

|  |
| --- |
| cd /etc/yum.repos.d/  mkdir backup  mv \*.repo ./backup |

2、下载阿里云源文件

|  |
| --- |
| wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-8.repo |

或

|  |
| --- |
| curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-8.repo |

3、检查以下是否成功

|  |
| --- |
| cat /etc/yum.repos.d/CentOS-Base.repo |

4、生成缓存

|  |
| --- |
| yum makecache |

### 开启命令自动补全

CentOS在最小化安装时，没有安装自动补全的包，需要手动安装：

|  |
| --- |
| yum -y install bash-completion |

安装好后，重新登陆即可（刷新bash环境）。

### CentOS8设置系统级别代理

**说明：**一般公司需要通过代理才能联网的情况下才需要设置系统级别代理。

1、编辑全局配置文件：*vim /etc/profile*

2、添加如下内容：

|  |
| --- |
| export http\_proxy="http://<user>:<password>@<proxy\_server>:<port>"  export https\_proxy="http://<user>:<password>@<proxy\_server>:<port>"  export ftp\_proxy=http://<user>:<password>@<proxy\_server>:<port>  export all\_proxy=http://<user>:<password>@<proxy\_server>:<port>  export HTTP\_PROXY="http://<user>:<password>@<proxy\_server>:<port>"  export HTTPS\_PROXY="http://<user>:<password>@<proxy\_server>:<port>"  export FTP\_PROXY=http://<user>:<password>@<proxy\_server>:<port>  export ALL\_PROXY=http://<user>:<password>@<proxy\_server>:<port> |

这里一口气全部设置，经试验这样设置后可以自动通过代理联网的应用有（目前只测试了这么多，后续测试到了继续往这里追加）：**yum、curl、wget**。

3、之后*source /etc/profile*或者重启CentOS8即可。

4、更过设置方法参看：2.1.3 设置代理

### CentOS7开机自启服务、脚本

**说明：**在CentOS7中，不建议再使用*rc.local*了，因此需要其它方法设置开机自启服务或脚本。

#### 添加开机自启服务

在CentOS7中添加开机自启服务非常方便，只需要两条命令，这里以*Jenkins*为例：

|  |
| --- |
| systemctl enable jenkins.service #设置jenkins服务为自启动服务  sysstemctl start jenkins.service #启动jenkins服务 |

**systemctl其它命令：**

|  |
| --- |
| systemctl daemon-reload #重载系统服务  systemctl enable \*.service #设置某服务开机启动  systemctl start \*.service #启动某服务  systemctl stop \*.service #停止某服务  systemctl reload \*.service #重启某服务 |

#### 添加开机自启脚本

1、以脚本*autostart.sh*为例，先编写*autostart.sh*脚本

2、将脚本移动到*/etc/rc.d/init.d*目录下

|  |
| --- |
| mv autostart.sh /etc/rc.d/init.d |

3、赋予脚本执行权限

|  |
| --- |
| chmod +x /etc/rc.d/init.d/autostart.sh |

4、加入开机启动

|  |
| --- |
| cd /etc/rc.d/init.d  chkconfig --add autostart.sh  chkconfig autostart.sh on |

5、查看是否添加成功

|  |
| --- |
| chkconfig --list |

6、重启验证

7、可能的问题

**1) bash: service: command not found**

执行*yum install initscripts -y*安装*service*指令

**2) service XXX does not support chkconfig**

必须把下面两行注释放在*/etc/init.d/autostart.sh*文件中

|  |
| --- |
| vim /etc/init.d/autostart.sh |

添加下面两句到*#!/bin/bash*之后：

|  |
| --- |
| #chkconfig: 2345 10 90  #description: autostart |

其中2345是默认启动级别，级别有0-6共7个级别。

* 等级0表示：表示关机
* 等级1表示：单用户模式
* 等级2表示：无网络连接的多用户命令行模式
* 等级3表示：有网络连接的多用户命令行模式
* 等级4表示：不可用
* 等级5表示：带图形界面的多用户模式
* 等级6表示：重新启动

10是启动优先级，90是停止优先级，优先级范围是0~100，数字越大，优先级越低。

上面的注释的意思是，zookeeper服务必须在运行级2，3，4，5下被启动或关闭，启动的优先级是10，关闭的优先级是90。

#### 自定义systemctl服务脚本

Centos7开机第一程序从*init*完全换成了*systemd*的启动方式，而*systemd*依靠*unit*的方式来控制开机服务，开机级别等功能。

Centos7的服务*systemctl* 脚本一般存放在：*/usr/lib/systemd*，目录下又有*user*和*system*之分：

* */usr/lib/systemd/system* #系统服务，开机不需要登录就能运行的程序（相当于开机自启）
* */usr/lib/systemd/user* #用户服务，需要登录后才能运行的程序

目录下又存在两种类型的文件：

* *\*.service* #服务级别*unit*
* *\*.target* #开机级别*unit*

CentOS7的每一个服务以*.service*结尾，一般会分为3部分：*[Unit]*、*[Service]*、*[Install]*，这里以x*xx.service*为例：*vim /usr/lib/systemd/system/xxx.service*

|  |
| --- |
| vim /usr/lib/systemd/system/xxx.service  [Unit] #主要是服务说明  Description=test #简单描述服务  After=network.target #描述服务类别，表示本服务需要在network服务启动后在启动  Before=xxx.service #表示需要在某些服务启动之前启动，After和Before字段只涉及启动顺序，不涉及依赖关系。  [Service] #核心区域  Type=forking #表示后台运行模式。  User=user #设置服务运行的用户  Group=user #设置服务运行的用户组  KillMode=control-group #定义systemd如何停止服务  PIDFile=/usr/local/test/test.pid #存放PID的绝对路径  Restart=no #定义服务进程退出后，systemd的重启方式，默认是不重启  ExecStart=/usr/local/test/bin/startup.sh #服务启动命令，命令需要绝对路径  PrivateTmp=true #表示给服务分配独立的临时空间  [Install]  WantedBy=multi-user.target #多用户 |

**字段说明：**

|  |
| --- |
| Type的类型:  simple(默认): #以ExecStart字段启动的进程为主进程  forking: #ExecStart字段以fork()方式启动，此时父进程将退出，子进程将成为主进程(后台 运行)。一般都设置为forking  oneshot: #类似于simple，但只执行一次，systemd会等它执行完，才启动其他服务  dbus: #类似于simple, 但会等待D-Bus信号后启动  notify: #类似于simple, 启动结束后会发出通知信号，然后systemd再启动其他服务  idle: #类似于simple，但是要等到其他任务都执行完，才会启动该服务。  EnvironmentFile:  指定配置文件，和连词号组合使用，可以避免配置文件不存在的异常。  Environment:  后面接多个不同的shell变量。  例如:  Environment=DATA\_DIR=/data/elk  Environment=LOG\_DIR=/var/log/elasticsearch  Environment=PID\_DIR=/var/run/elasticsearch  EnvironmentFile=-/etc/sysconfig/elasticsearch  连词号(-):  在所有启动设置之前，添加的变量字段，都可以加上连词号，表示抑制错误，即发生错误时，不影响其他命 令的执行。  比如EnviromentFile=-/etc/sysconfig/xxx表示即使文件不存在，也不会抛异常  KillMode的类型:  control-group(默认): #当前控制组里的所有子进程，都会被杀掉  process: #只杀主进程  mixed: #主进程将收到SIGTERM信号，子进程收到SIGKILL信号  none: #没有进程会被杀掉，只是执行服务的stop命令  Restart的类型:  no(默认值): #退出后无操作  on-success: #只有正常退出时(退出状态码为0),才会重启  on-failure: # 非正常退出时，重启，包括被信号终止和超时等  on-abnormal: # 只有被信号终止或超时，才会重启  on-abort: # 只有在收到没有捕捉到的信号终止时，才会重启  on-watchdog: # 超时退出时，才会重启  always: # 不管什么退出原因，都会重启  #对于守护进程，推荐用on-failure  RestartSec字段:  表示systemd重启服务之前，需要等待的秒数:RestartSec:30  Exec\*字段:  #Exec\*后面接的命令，仅接受“指令 参数 参数..”格式，不能接受 <> | & 等特殊字符，很多bash语法也不支持。如果想支持bash语法，需要设置Tyep=oneshot  ExecStart: #启动服务时执行的命令  ExecReload: #重启服务时执行的命令  ExecStop: #停止服务时执行的命令  ExecStartPre: #启动服务前执行的命令  ExecStartPost: #启动服务后执行的命令  ExecStopPost: #停止服务后执行的命令  WantedBy字段:  multi-user.target: #表示多用户命令行状态，这个设置很重要  graphical.target: #表示图形用户状体，它依赖于multi-user.target |

## Manjaro专区

### Win + Manjaro 双系统、双硬盘安装方法、正确引导系统方法 、黑屏解决方法

**1) 前言**

**本教程只涉及Win+Manjaro双系统、双硬盘安装过程中的核心要点，不涉及具体步骤，不注意这些要点，安装之后是进不去Manjaro系统的。**

详细的安装步骤网上已经有很多了，这里不再给出，可以参看以下文章，熟悉流程：

* [安装Manjaro双系统](https://mogeko.me/2019/059/)
* [Manjaro开机黑屏卡住\_显卡驱动问题解决及配置源和搜狗输入法安装](https://blog.csdn.net/Umbrella2B/article/details/84258951)

**你可能遇到的问题：**

你有一个固态硬盘安装了Win10系统，还有一个机械硬盘划分出了100G空间，按照上述文章中所说的步骤，把Manjaro安装在了机械硬盘的100G空间上，并用EasyBCD引导，可是进入Manjaro系统黑屏，或者只有一个下划线闪烁。

我也遇到了上述黑屏问题，直到看了这篇文章才知道问题出在了哪里，[Ubuntu16.04与Win10双系统双硬盘安装图解](https://blog.csdn.net/fesdgasdgasdg/article/details/54183577)。问题就是：双硬盘分别安装了两个系统，这没有问题，但是两个系统的引导分区一定要在一块硬盘上。这篇作者说的是两个系统一定要安装在一个硬盘上，这是不对的，只需要保证两个系统的引导分区在一个硬盘上即可，系统可以分别放在两个硬盘上。

**安装环境：**

* DELL灵越笔记本5420
* 集成显卡
* 英伟达独立显卡
* 120G固态
* 500G机械硬盘

**实现目标：**

Win10装在固态上，500G硬盘划出100G空间安装Manjaro系统，100G足够了，安装完Manjaro系统，在Manjaro系统里边是可以访问120G固态和500G机械硬盘的。

**2) 安装过程要点**

**① GRUP安装界面**

电脑含集成显卡和英伟达独立显卡，安装系统的时候要选择集成显卡，要不然容易开机进不去，系统安装完成，进入系统正式使用后再安装英伟达独立显卡驱动。

这一步详细的设置步骤可以参看上边两篇文章，或者其它文章，这里只讲要点。

**步骤：**

1、引导盘开机看到启动菜单的时候，用方向键移到BOOT那一栏，长按键盘*E*键进入编辑页面；

2、将*driver=free*改成*driver=intel*，并在其后面加上*xdriver=mesaacpi\_osi=!acpi\_osi="Windows2009"*。解释：*free*表示开源显卡驱动，*nofree*表示专有显卡驱动，例如英伟达显卡；*intel*表示集成显卡驱动；后边那一段大致意思是告诉BIOS我是Win7，别闹了，好好工作；

3、然后按*Ctrl+X*或者*F10*启动。

**② 分区界面**

**要点：**双硬盘环境下，两个系统的开机引导一定要在一个盘上，不能一个在固态硬盘上，一个在机械硬盘上，这样开机是进不去Manjaro系统的。

**解决方案：**

1、原来安装Win10的固态硬盘划出1G的空间，安装Manjaro系统引导，即Manjaro */boot*的安装位置；

2、Manjaro系统安装在机械硬盘100G的空间上，包括Manjaro系统根目录*/*，交换分区*linuxswap*，用户主目录*/home*等；

3、安装完成后进入Win10，安装EasyBCD添加Manjaro系统的引导，引导位置是120G固态硬盘上划分出1G装Manjaro */boot*的位置，再重启即可正确引导进入Manjaro系统。

**分区界面：**

1、把Manjaro的*/boot*分区安装在固态硬盘换分出的1G空间上，标记设置为*boot*，这一步是重点，一定要弄对；

2、其它系统分区，包括Manjaro系统根目录*/*，交换分区*linuxswap*，用户主目录*/home*等安装在机械硬盘100G的空间上。这里到底是否分区，每个分区分多少空间，参看其它文章，众说纷纭，但这都不知重点，怎么分都不会影响你是否可以进入Manjaro系统；

3、最下边的引导安装位置，选择*/boot*，即固态硬盘上划分出安装Manjaro */boot*分区的1G空间，这一步是重点，一定要弄对。

### 添加archlinux镜像源

**1) 步骤一**

向*/etc/pacman.d/mirrorlist*中添加国内镜像地址。

**方法1：自动添加**

1、输入如下命令查看国内镜像源，并按质量排序：*sudo pacman-mirrors -i -c China -m rank*，之后会弹出一个窗口，可以选择想要的镜像源，选择确定后会自动导入*/etc/pacman.d/mirrorlist*配置文件中。

**方法2：手动添加**

直接在*etc/pacman.d/mirrorlist*文件后边添加如下内容，这些是根据pacman-mirrors命令选出以及网友推荐的比较优质的，也可以添加其它的：

|  |
| --- |
| ## Country : China（中科大）  Server = https://mirrors.ustc.edu.cn/manjaro/stable/$repo/$arch  ## Country : China（清华）  Server = https://mirrors.tuna.tsinghua.edu.cn/manjaro/stable/$repo/$arch |

步骤一是必须的，要不然运行**步骤三**会提示找不到仓库。

**2) 步骤二**

编辑pacman配置文件

1、打开pacman配置文件

|  |
| --- |
| sudo vim /etc/pacman.conf #打开 pacman 配置文件 |

2、在文件末尾添加以下内容：

|  |
| --- |
| [archlinuxcn]  SigLevel = Optional TrustedOnly  Server = https://mirrors.tuna.tsinghua.edu.cn/archlinuxcn/$arch |

**解释：**

* *SigLevel*可以设置为Nerver来屏蔽验证要安装的软件包；
* *Server = https://mirrors.tuna.tsinghua.edu.cn/archlinuxcn/$arch*是清华大学的镜像源，也可以从下边的列表中选出一个其它的。

**官方镜像源：**

|  |
| --- |
| [archlinuxcn]  Server = https://repo.archlinuxcn.org/$arch  #### 浙江大学 (浙江杭州) (ipv4, ipv6, http, https)  #### Added: 2017-06-05  [archlinuxcn]  Server = https://mirrors.zju.edu.cn/archlinuxcn/$arch  #### 中国科学技术大学 (ipv4, ipv6, http, https)  [archlinuxcn]  Server = https://mirrors.ustc.edu.cn/archlinuxcn/$arch  #### 清华大学 (ipv4, ipv6, http, https)  [archlinuxcn]  Server = https://mirrors.tuna.tsinghua.edu.cn/archlinuxcn/$arch  #### Our main server (ipv4, ipv6, http, https)  #### Our main server located in Netherlands  [archlinuxcn]  Server = https://repo.archlinuxcn.org/$arch  #### xTom (Hong Kong server) (Hong Kong) (ipv4, ipv6, http, https)  #### Added: 2017-09-18  #### xTom Hong Kong Mirror  [archlinuxcn]  Server = https://mirror.xtom.com.hk/archlinuxcn/$arch  #### xTom (US server) (US) (ipv4, ipv6, http, https)  #### Added: 2019-02-19  #### xTom US Mirror  [archlinuxcn]  Server = https://mirror.xtom.com/archlinuxcn/$arch  #### Open Computing Facility, UC Berkeley (Berkeley, CA, United States) (ipv4, ipv6, http, https)  #### Added: 2019-02-19  [archlinuxcn]  Server = https://mirrors.ocf.berkeley.edu/archlinuxcn/$arch  #### 网易 (ipv4, http, https)  [archlinuxcn]  Server = https://mirrors.163.com/archlinux-cn/$arch  #### 重庆大学 (ipv4, http, https)  [archlinuxcn]  Server = https://mirrors.cqu.edu.cn/archlinuxcn/$arch  #### SJTUG 软件源镜像服务 (ipv4, https)  #### Added: 2018-05-21  [archlinuxcn]  Server = https://mirrors.sjtug.sjtu.edu.cn/archlinux-cn/$arch  #### 莞工 GNU/Linux 协会 开源软件镜像站 (ipv4, http, https)  #### Added: 2018-11-03  [archlinuxcn]  Server = https://mirrors.dgut.edu.cn/archlinuxcn/$arch  #### 腾讯云 (ipv4, https)  #### Added: 2018-11-23  [archlinuxcn]  Server = https://mirrors.cloud.tencent.com/archlinuxcn/$arch |

**3) 步骤三**

同步软件仓库，建立索引

|  |
| --- |
| sudo pacman -Syy |

**4) 步骤四**

安装archlinuxcn-keyring包以导入GPG key

|  |
| --- |
| sudo pacman -S archlinux-keyring |

### 中文字体渲染美化+去模糊

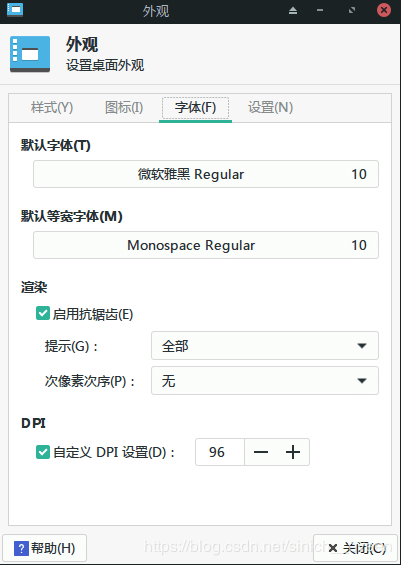
**步骤：**

1**、**解压**freetype2-ultimate5.tar.gz**和**lib32-freetype2-ultimate5.tar.gz**两个压缩包，分别按照里面的《安装方法》操作；

2**、**解压安装**lulinux\_fontsConf\_181226.tar.gz**，按里面的安装说明操作；

3**、**开启字体渲染：

打开**外观**，更改字体：



默认字体：微软雅黑 Regular

默认等宽自体：Monospace Regular

启用抗锯齿 > 提示：全部

4**、**更改Qt5字体

**Qt5设置 > 字体**



5、附件

32-F-Manjaro中文字体渲染+去模糊

## WSL专区

### Xshell连接WSL Ubuntu 18.04 LTS

#### 方法1（此方法可能不行）

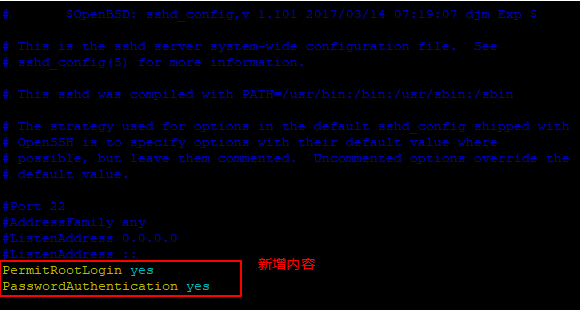
1、卸载重装一遍ssh服务（自带的不好用）

|  |
| --- |
| sudo apt-get remove openssh-server  sudo apt-get install openssh-server |

2、编辑*sshd\_config*文件

|  |
| --- |
| sudo vim /etc/ssh/sshd\_config |

在此位置添加如下内容：



|  |
| --- |
| PermitRootLogin yes #如果需要用 root 直接登录系统则此处改为 yes  PasswordAuthentication yes #表示使用帐号密码方式登录 |

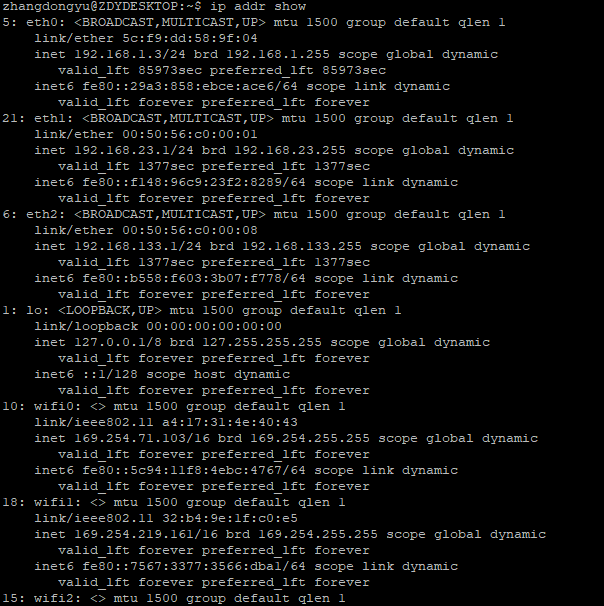
为了安全可以把默认端口*Port 22*改为其它值。

3、启动ssh服务

|  |
| --- |
| sudo service ssh start |

4、查看WSL IP地址

|  |
| --- |
| ip addr show  #或  ifconfig |



5、Putty/Xshell中输入WSL IP和端口号（默认：*22*）连接。本例中既可以用WSL IP *192.168.1.3*连接，也可以用本机回路IP *127.0.0.1*连接。推荐使用本机回路IP连接，因为不会变，而WSL IP可能随着Win10系统重启而改变。如果使用*127.0.0.1*无法连接，参看**方法2**。

6、配置ssh服务随Win10自启动

上边虽然让Xshell连接上了WSL，但是每次Win10重启后，ssh服务不会自动运行，需要打开WSL（例如：Ubuntu-18.04 LTS）终端启动ssh服务，然后再打开Xshell连接，十分麻烦。可以配置让ssh随Win启动而启动。参看：2.4.3 WSL2服务自启动

#### 方法2（此方法一定可行）

参考网址：

* [WSL2 Set static ip? #4210](https://github.com/microsoft/WSL/issues/4210)
* [BAT以管理员运行且不弹黑窗口二者不能兼得，真的很难吗！](http://www.bathome.net/thread-49107-1-1.html)
* [怎样自动以管理员身份运行bat文件?](https://www.zhihu.com/question/34541107)

1、先按照**方法1**操作完1~3步

2、给WSL Ubuntu和Win10添加固定ip

**1) 临时生效：**

在Windows 10中，以管理员权限运行cmd或Windows Powershell，然后执行以下两个命令：

|  |
| --- |
| #在Ubuntu中添加IP地址192.168.50.16，名为eth0：1  wsl -d Ubuntu-18.04 -u root ip addr add 192.168.50.16/24 broadcast 192.168.50.255 dev eth0 label eth0:1  #在Win10中添加IP地址192.168.50.88  netsh interface ip add address "vEthernet (WSL)" 192.168.50.88 255.255.255.0 |

然后，可用*192.168.50.16*访问Ubuntu，*192.168.50.88*访问Win10。

**2) 开机自动设置：**

**①方法1：**

a、「开始 > 运行 > 输入：shell:startup」

b、在此**启动**目录下建立一个*wsl\_ip.bat*脚本，添加如下内容：

|  |
| --- |
| @echo off  :: 获取管理员权限  setlocal  set uac=~uac\_permission\_tmp\_%random%  md "%SystemRoot%\system32\%uac%" 2>nul  if %errorlevel%==0 ( rd "%SystemRoot%\system32\%uac%" >nul 2>nul ) else (  echo set uac = CreateObject^("Shell.Application"^)>"%temp%\%uac%.vbs"  echo uac.ShellExecute "%~s0","","","runas",1 >>"%temp%\%uac%.vbs"  echo WScript.Quit >>"%temp%\%uac%.vbs"  "%temp%\%uac%.vbs" /f  del /f /q "%temp%\%uac%.vbs" & exit )  endlocal  :: 给WSL Ubuntu和Win10添加固定ip  wsl -d Ubuntu-18.04 -u root ip addr add 192.168.50.16/24 broadcast 192.168.50.255 dev eth0 label eth0:1  netsh interface ip add address "vEthernet (WSL)" 192.168.50.88 255.255.255.0 |

因为*netsh interface ip add address "vEthernet (WSL)" 192.168.50.88 255.255.255.0*这条命令必须以管理员身份运行，所以此脚本的上边一段代码是首先获取管理员运行权限，然后才开始执行设置ip相关的命令；

c、因为此脚本放在**启动**目录下，所以Win10启动的时候会自动运行此脚本设置ip，就可以用Xshell通过*192.168.50.16*访问WSL Ubuntu了；

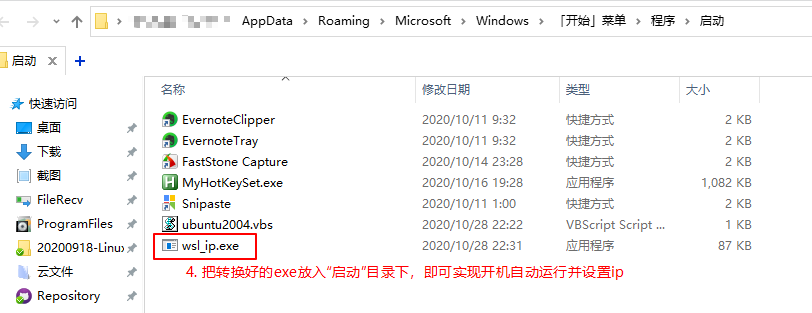
d、此方法缺点：每次开机运行此脚本时都会弹出一个黑窗然后自动关闭，不是很优雅。

**②方法2:**

通过**BatToExeConverter.exe**把上述脚本转换成**wsl\_ip.exe**，然后放入**启动**目录下，开机之后就不会弹出黑窗了，比较完美。

**操作步骤：**





5、这里提供制作好的**wsl\_ip.exe**。

6、配置ssh开机自启参看**方法1**的步骤6。

附件：32-S-wsl\_ip.exe

附件：32-S-BatToExeConverter.zip

### WSL1升级为WSL2

[Windows Subsystem for Linux Installation Guide for Windows 10](https://docs.microsoft.com/en-us/windows/wsl/install-win10#step-4---download-the-linux-kernel-update-package)

[适用于Linux的Windows子系统安装指南(Windows 10)](https://docs.microsoft.com/zh-cn/windows/wsl/install-win10#step-4---download-the-linux-kernel-update-package)

### WSL2服务自启动

**教程参考来源：**[WSL服务自动启动的正确方法](https://zhuanlan.zhihu.com/p/47733615)

进入任意WSL发行版中，创建并编辑文件：*vim /etc/init.wsl*

|  |
| --- |
| #!/bin/sh  /etc/init.d/cron $1  /etc/init.d/ssh $1  /etc/init.d/supervisor $1 |

里面调用了我们希望启动的三个服务的启动脚本，设置*/etc/init.wsl*权限*777*，这时候可以通过：

|  |
| --- |
| sudo /etc/init.wsl [start|stop|restart] |

来启停我们需要的服务，在Windows中，**开始 > 运行**，输入：

|  |
| --- |
| shell:startup |

在打开的文件夹中按照你WSL使用的Linux发行版创建启动脚本，比如创建的*Debian.vbs*文件：

|  |
| --- |
| Set ws = CreateObject("Wscript.Shell")  ws.run "wsl -d debian -u root /etc/init.wsl start", vbhide |

这个脚本就会在你登陆的时候自动在名字为*debian*的wsl发行版中执行*/etc/init.wsl*启动我们的服务了，如果你用的是Ubuntu-18.04 LTS的发行版，那么修改上面脚本里的*debian*为*Ubuntu-18.04*，并创建*ubuntu1804.vbs*：

|  |
| --- |
| Set ws = CreateObject("Wscript.Shell")  ws.run "wsl -d Ubuntu-18.04 -u root /etc/init.wsl start", vbhide |

而如果你不知道自己的WSL发行版叫做什么名字，可以用*wsl -l*来查看。

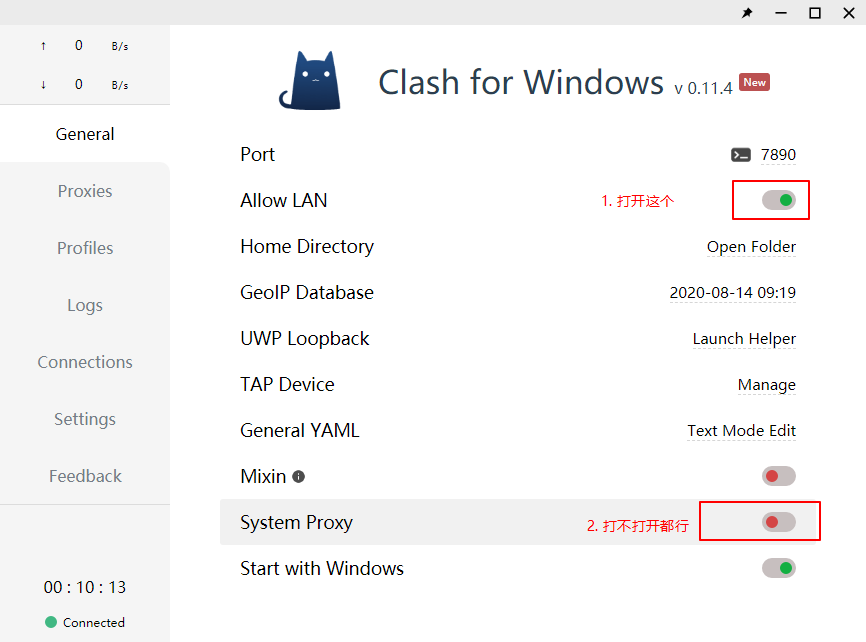
WSL中有很多有用的服务，你可以按需删改*/etc/init.wsl*，但没必要塞很多东西进去影响你的启动速度，比如*mysql/mongodb*这些重度服务，可以需要的时候再启动，用完就停了。

### WSL2 Ubuntu安装桌面

* [WSL2 下 Ubuntu 18.04 安装配置和使用 Gnome 桌面](https://www.chengxuzhilu.com/2407.html#geng_xin_xi_tong)（已剪藏到印象笔记）
* [WSL2 下让 Ubuntu20.04 安装 Gnome 桌面](https://www.chengxuzhilu.com/2395.html)（已剪藏到印象笔记）

### WSL2通过Clash for Windows使用Windows代理

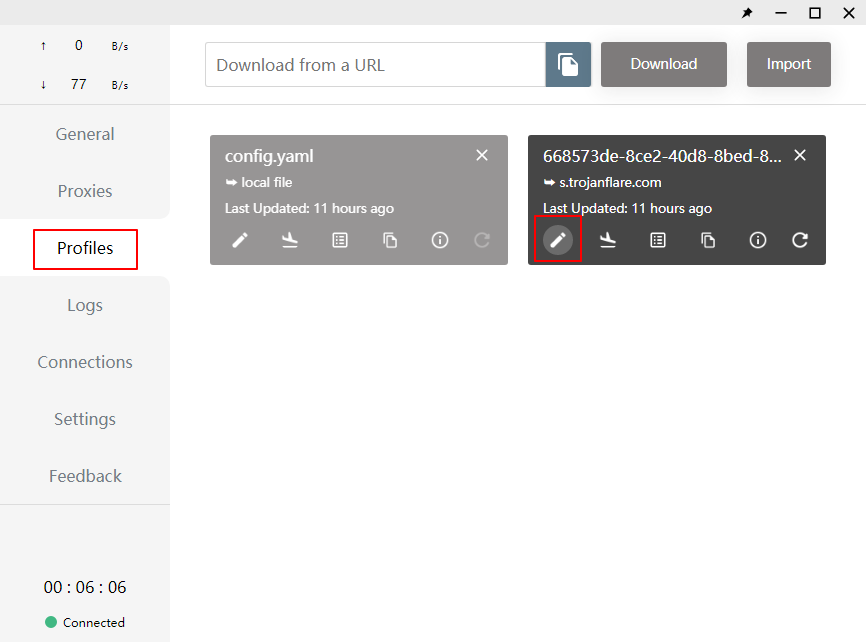
1、Clash for Windows打开“Allow LAN”选项



这里不打开“System Proxy”为例，此时Win10系统代理如下图所示：



2、从配置文件中查看端口：





两个端口是不一样的。配置文件参数具体解释参看：[Clash for Windows Book](https://docs.cfw.lbyczf.com/contents/ui/general.html)的“常规General”章节。

3、打开shell终端，输入如下命令：

|  |
| --- |
| export hostip=$(cat /etc/resolv.conf |grep -oP '(?<=nameserver\ ).\*')  export https\_proxy="http://${hostip}:7890"  export http\_proxy="http://${hostip}:7890" |

这里只设置了*https\_proxy*、*http\_proxy*，没有设置*all\_proxy*等，并且也只使用了*http*协议代理，没有设置*sock5*协议代理，可以根据需要自己更改，但是需要注意，本例中*http*协议端口是*7890*，而如果要设置*sock5*协议，那么端口需要改为*7891*。具体设置方法参看：设置HTTP、HTTPS、FTP、SOCK5代理

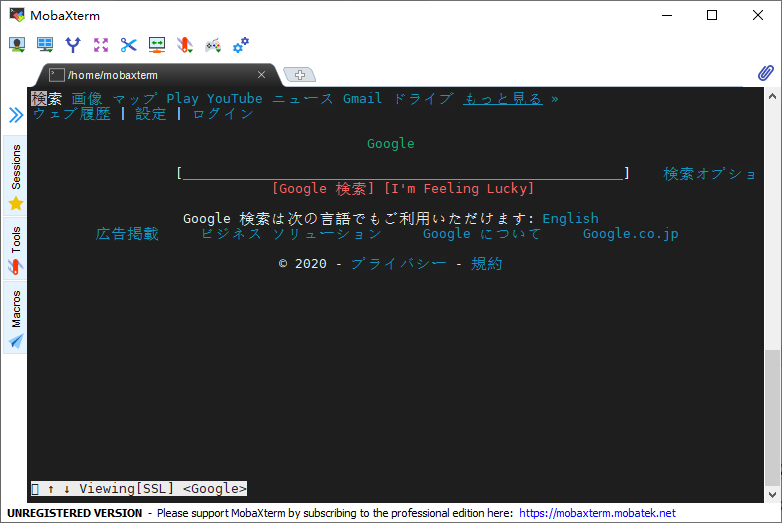
4、安装w3m

|  |
| --- |
| sudo apt install w3m |

5、用w3m进行测试

|  |
| --- |
| w3m www.google.com |

如果成功，运行完该命令只有会在shell中出现如下画面：



6、一劳永逸，可以在*~/.bashrc*添加第3步的四条命令。

### 修改WSL默认为root登录

打开Windows PowerShell，输入如下命令：

|  |
| --- |
| <wsl\_edition> config --default-user root  #e.g. ubuntu1804.exe config --default-user root |

### WSL和Proxifier共存

打开cmd输入如下命令：

|  |
| --- |
| D:\下载\NoLsp.exe C:\windows\system32\wsl.exe |

其中**NoLsp.exe**和**wsl.exe**替换为真实路径。

附件：32-S-NoLsp.zip

### 参考的对象类型不支持尝试的操作

打开Windows PowerShell，输入如下命令，然后重启即可：

|  |
| --- |
| netsh winsock reset |

## 通用专区

### .bashrc自定义命令

|  |
| --- |
| #START  #脚本  source /root/functions/z-apt-uninstall.sh  source /root/functions/z-man-search.sh  source /root/functions/z-trash.sh  #切换路径  alias z-dir-repository="cd /root/onedrive/Repository"  alias z-dir-cpp="cd /root/onedrive/Repository/C\_PLUS\_PLUS"  alias z-dir-python="cd /root/onedrive/Repository/PYTHON"  alias z-dir-download="cd /root/download"  alias z-dir-linux-manual="cd /root/link/linux-manual"  alias z-git-config-diff-word="touch .gitattributes && echo \\*.docx diff=word > .gitattributes && git config diff.word.textconv docx2txt"  #Git  alias z-git-log="git log --oneline --graph"  #系统相关  PS1="[\t \W]\$ "  export HISTTIMEFORMAT="%F %H:%M:%S "  #命令  alias z-vim-bashrc="vim ~/.bashrc"  alias z-cat-bashrc="cat ~/.bashrc"  alias z-source-bashrc="source ~/.bashrc"  alias z-trash-clean="rm -rf /root/.trash/\*"  #END |

### 添加可执行文件、C/C++头文件、动态库、静态库路径

对所有用户有效在，则在*/etc/profile*添加：

|  |
| --- |
| #可执行文件路径  export PATH =$PATH:$HOME/bin  #gcc找到头文件的路径  C\_INCLUDE\_PATH=$C\_INCLUDE\_PATH:/usr/include/libxml2:/MyLib  export C\_INCLUDE\_PATH  #g++找到头文件的路径  CPLUS\_INCLUDE\_PATH=$CPLUS\_INCLUDE\_PATH:/usr/include/libxml2:/MyLib  export CPLUS\_INCLUDE\_PATH  #找到动态链接库的路径  LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/MyLib  export LD\_LIBRARY\_PATH  #找到静态库的路径  LIBRARY\_PATH=$LIBRARY\_PATH:/MyLib  export LIBRARY\_PATH |

*reboot*后就可以使用了。

如果只对当前用户有效，则在Home目录下的*.bashrc*或*.bash\_profile*里增加以上内容。

### 为命令历史记录(history)启用时间戳

当前shell有效：*export HISTTIMEFORMAT="%F %H:%M:%S "*

永久有效：写入*.bashrc*即可

时间格式：可以查看*date*命令的man page：*man date*

### 取消 sudo 密码

1、*sudo visudo*

2、将*%sudo ALL=(ALL:ALL) ALL*改为*%sudo ALL=(ALL:ALL) NOPASSWD:ALL*

### 为普通用户添加 sudo 权限

1、编辑配置文件*sudo gedit /etc/sudoers*

2、在*root ALL=(ALL) ALL*的下一行添加一行*<username> ALL=(ALL) ALL*

### 减少默认的 grub 载入时间

1、*sudo gedit /etc/default/grub &*

2、将*GRUB\_TIMEOUT=10*改为*GRUB\_TIMEOUT=<n>*，这将改变启动时间为*n*秒

3、*sudo update-grub*使配置生效

### 鼠标右键新建文档

在**主目录**的**模板**文件夹中放一个对应类型的文件，鼠标右键即可新建该类型文档。

### 中文文件夹名字改为英文

1**、**编辑*~/.config/user-dirs.dirs*文件

|  |
| --- |
| gedit ~/.config/user-dirs.dirs |

2**、**修改文件内容为：

|  |
| --- |
| XDG\_DESKTOP\_DIR="$HOME/Desktop"  XDG\_DOWNLOAD\_DIR="$HOME/Downloads"  XDG\_TEMPLATES\_DIR="$HOME/Templates"  XDG\_PUBLICSHARE\_DIR="$HOME/Public"  XDG\_DOCUMENTS\_DIR="$HOME/Documents"  XDG\_MUSIC\_DIR="$HOME/Music"  XDG\_PICTURES\_DIR="$HOME/Pictures"  XDG\_VIDEOS\_DIR="$HOME/Videos" |

3**、**在**文件管理器**应用里边把现有的中文名字改为对应的英文名字，**桌面**文件夹此时被占用，会再生成一个**Desktop**文件夹。

4**、**注销再登陆，删除**桌面**文件夹即可。

### 在Windows PowerShell中重启WSL Ubuntu 18.04 LTS

**命令：**

|  |
| --- |
| wsl -t Ubuntu-18.04 #关闭 WSL Ubuntu  ubuntu1804.exe #启动WSL Ubuntu |

可以运行*wsl -h*查看wsl命令的用法。

**为了方便，可以在PowerShell中创建Alias命令：**

Alias命令创建方法参考：[Windows PowerShell 配置文件](https://forsenergy.com/zh-cn/windowspowershellhelp/html/9c82251c-6f0d-416a-9c3c-77838218531b.htm)和[PowerShell设置命令别名Alias系统美化](https://segmentfault.com/a/1190000015928399)

PowerShell中禁止执行脚本问题参看：[PowerShell因为在此系统中禁止执行脚本解决方法](https://www.cnblogs.com/zhaozhan/archive/2012/06/01/2529384.html)

在PowerShell中执行*.\notepad.exe $PROFILE*，在打开的文件中添加如下内容：

|  |
| --- |
| function restartUbuntu{  wsl -t Ubuntu-18.04  ubuntu1804.exe  }  Set-Alias z-restart-ubuntu restartUbuntu |

### Hyper-V Linux挂载Win10目录

1、**找到要共享的文件夹>右击文件夹/盘>选择属性>点开共享选项卡>选择高级共享>勾选共享此文件夹>点开权限>点击添加>可以指定一个用户或者直接用Everyone>权限选择完全控制（读和写）>确定保存**；

2、转到linux下，运行如下命令挂载Win10共享目录：

|  |
| --- |
| mkdir /mnt/<dir\_name>  mount -o username=<user\_name>,password=<password> //<host\_ip>/<file\_path> /mnt/<dir\_name> |

注意，这里只是临时有效；

3、开机自动挂载

|  |
| --- |
| vim /ets/fatab |

末尾添加如下内容，即可实现开机自动挂载：

|  |
| --- |
| //<host\_ip>/<file\_path> /mnt/<dir\_name> cifs username=<user\_name>,password=<password> 0 0 |

下次Linux启动时就会自动挂载。

# 知识点合集

## Ubuntu专区

### PPA(Personal Package Archive)

[PPA完全指南：如何在Ubuntu系统中使用PPA？](https://www.sysgeek.cn/ubuntu-ppa/)

### apt与apt-get命令的区别与解释

[Ubuntu中apt与apt-get命令的区别与解释](https://www.sysgeek.cn/apt-vs-apt-get/)

### Ubuntu软件包安装管理体系

[Ubuntu软件安装管理之——dpkg与apt-\*详解](https://segmentfault.com/a/1190000011463440)

## CentOS专区

### CentOS软件包安装管理体系

[Linux系统中软件的“四”种安装原理详解：源码包安装、RPM二进制安装、YUM在线安装、脚本安装包](https://segmentfault.com/a/1190000011325357)

[RPM与YUM详解](https://segmentfault.com/a/1190000011200461)

## Manjaro专区

## 通用专区

### 主流软件包管理体系



### 源码安装

[源码安装详解](https://segmentfault.com/a/1190000011200004?share_user=1030000007255638)

### bashrc、profile、environment区别

1、在Ubuntu中有如下几个文件可以设置环境变量

* */etc/profile*：在登录时，操作系统定制用户环境时使用的第一个文件，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。
* */etc/environment*：在登录时操作系统使用的第二个文件，系统在读取你自己的*profile*前，设置环境文件的环境变量。
* *~/.profile*：在登录时用到的第三个文件是*.profile*文件，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，它设置一些环境变量，执行用户的*.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。
* *~/.bashrc：*该文件包含专用于你的bash shell的bash信息，当登录时以及每次打开新的shell时，该该文件被读取。

2、通常设置环境变量有三种方法

① 临时设置：*export PATH=/home/yan/share/usr/local/arm/3.4.1/bin:$PATH*

② 当前用户的全局设置：

|  |
| --- |
| gedit ~/.bashrc  export PATH=/home/yan/share/usr/local/arm/3.4.1/bin:$PATH #末尾添加  source .bashrc #使生效 |

③ 所有用户的全局设置

|  |
| --- |
| gedit /etc/profile  export PATH=/home/yan/share/usr/local/arm/3.4.1/bin:$PATH #末尾添加  source profile |

④ 测试是否添加成功：*echo $PATH*

### Linux .conf .d含义

这个*.d*是表示目录(directory)的意思。

早期unix中很多应用软件通常都只用一个独立的配置文件，如*fstab*等。随着应用越来越复杂，软件中需要配置的项目越来越多，单个配置文件开始显得复杂无比，难以操作。这时很多软件就逐渐开始将配置项放入多个文件。使用多个配置文件更容易分配操作权限，内容也可以更专一，因此更适合模块化管理。但这也造成了*/etc/*目录下的文件急剧增加，且文件名冲突现象越来越严重。

因此unix 中约定系统软件除创建一个（特例下可以多个，如*rc*簇)传统的全局配置文件外，将附属的其它配置文件保存到同名的目录中，区别是在名称后面加*.d*后缀，同时全局配置文件后面也加上后缀*.conf*，这样就使得配置文件更容易进行模块化管理，同时兼容了unix的老习惯：直接操作*/etc/*下的配置文件，而不是操作*/etc/*下子目录中的配置文件。

对于安装在unix中的应用软件来说，它们的配置文件除自己使用外，很少会由其它应用来读取或修改，因此应用软件完全可以由自己来管理配置文件，所以应用类软件通常直接在*/etc/*下创建一个子目录用来保存所有的配置文件，而不再创建传统的全局配置文件。

所以，我们现在能在*/etc/*看到的就是这三种不同的配置文件组织方式下的文件和子目录。分别是：传统的单一配置文件(都是比较早期开发的软件)、系统软件的全局配置文件(.conf)+附属配置文件(.d)、以及应用软件的配置文件（存放在子目录且子目录名不加*.d*）。

另外，unix中还有一个约定，凡是守护进程的执行文件名后会加上*d*，这个*d*前不带"*.*"号，代表的才是daemon（守护进程）的意思，通常存放在*/usr/sbin/*下。

### Linux命令格式

linux shell命令通常可以通过*-h*或*--help*来打印帮助说明，或者通过*man*命令来查看帮助，有时候我们也会给自己的程序写简单的帮助说明，其实帮助说明格式是有规律可循的。

**帮助示例**

下面是*git reset*命令的帮助说明，通过*man git-reset*可以查看：

|  |
| --- |
| git reset [-q] [<tree-ish>] [--] <paths>...  git reset (--patch | -p) [<tree-ish>] [--] [<paths>...]  git reset [--soft | --mixed | --hard | --merge | --keep] [-q] [<commit>] |

**对于命令和参数大致有如下几种类型：**

|  |
| --- |
| 没有任何修饰符参数 : 原生参数  <> : 占位参数  [] : 可选组合  () : 必选组合  | : 互斥参数  ... : 可重复指定前一个参数  -- : 标记后续参数类型 |

**参数类型解读**

1、原生参数

说明文档里的字符即为命令需要使用的字符，比如以上命令的：

|  |
| --- |
| git reset |

这种参数在使用时必需指定，且和说明文档里的一致。

2、占位参数

表示方式：*< >*

和原生参数类似，都是必需指定的，只不过占位参数的实际字符是在使用时指定的，同时为了方便阅读会用一个描述词汇来表示，并以*< >*包围，比如：

|  |
| --- |
| <paths> |

表示路径，使用时可以指定为具体的路径，而*paths*只是起一个说明作用，有些帮助说明里也会用大写来表示占位参数，比如将以上参数说明写成*PATHS*。

3、可选组合

表示方式：*[]*

括号里的参数为可选参数，比如usage第二个里面的*[-q]*，则*-q*为可选参数。

可选项和占位参数也可以同时使用，如：

|  |
| --- |
| [<commit>] |

表示该参数可以指定某次提交，也可以不指定。

4、必选组合

表示方式：*()*

括号里的参数必需指定，通常里面会是一些互斥参数，比如：

|  |
| --- |
| (--patch | -p) |

表示*--patch*和*-p*这两个参数必需指定一个。

5、互斥参数

表示方式：*|*

互斥参数一般都在*()*和*[]*里，表示该参数只能指定其中一个，比如：

|  |
| --- |
| [--mixed | --soft | --hard | --merge | --keep] |

6、重复参数

表示方式：*...*

表示前一个参数可以被指定多个，比如：

|  |
| --- |
| <paths>... |

*<paths>*是一个占位参数，使用时必需指定为路径，*...*并表示可以指定多个路径。重复参数的一个典型使用场景就是移动文件，将多个文件移动到一个目录下，比如如下命令：

|  |
| --- |
| git mv [<options>] <source>... <destination> |

我们可以这样使用：

|  |
| --- |
| git mv -f a.cpp b.py dir |

此时*options*对应为*-f*参数，*source*对应为*a.cpp b.py*，*destination*对应为*dir*。

7、标记后续参数类型

表示方式：*--*

表示后续参数的某种类型，比如这里如果使用如下命令：

|  |
| --- |
| git reset -p -- xx |

对比第一个命令，这里的*xx*对应的应该是*<paths>*参数，当我们指定*--*之后，则git会认为*xx*就是一个路径，那怕它是特殊符号或者路径并不存在。这是shell命令的一个通用方式，比如我们有一个文件名为*-h*，如果想删除这个文件，执行：

|  |
| --- |
| rm -h |

肯定是无法删除的，因为这时-h会被认为是rm的一个参数选项，应该使用：

|  |
| --- |
| rm -- -h |

这时shell会将*-h*解释为一个文件名传递给*rm*命令。

**解读实战**

最后来解释一个比较复杂的帮助说明：

|  |
| --- |
| git cat-file (-t [--allow-unknown-type]|-s [--allow-unknown-type]|-e|-p|<type>|--textconv) <object> |

该命令参数由四个部分，其中git和*cat-file*为原生参数，*()*里的为必选组合，*<object>*为占位参数

组合又由6部分组成，为互斥关系：

|  |
| --- |
| -t [--allow-unknown-type]  -s [--allow-unknown-type]  -e  -p  <type>  --textconv |

因此该命令的帮助说明可以拆分如下：

|  |
| --- |
| git cat-file -t <object>  git cat-file -t --allow-unknown-type <object>  git cat-file -s <object>  git cat-file -s --allow-unknown-type <object>  git cat-file -e <object>  git cat-file -p <object>  git cat-file <type> <object>  git cat-file --textconv <object> |

### /etc/fstab、/etc/mtab、/proc/mounts 文件区别

**1) /etc/fstab**

*/etc/fstab*是开机自动挂载的配置文件，在开机时起作用。相当于启动linux的时候，自动使用检查分区的*fsck*命令和挂载分区的*mount*命令，检查分区和挂载分区都是根据*/etc/fstab*中记录的相关信息进行的。

**2) /etc/mtab**

*/etc/mtab*是当前的分区挂载情况，记录的是当前系统已挂载的分区。每次挂载/卸载分区时会更新*/etc/mtab*文件中的信息。执行*mount*命令会改变*/etc/mtab*的信息。

**3) /proc/mounts**

这个文件是*/proc/self/mounts*的软链接，*/proc*下面的文件都是保存在内存中的，是内核自动生成的。所以*/proc/mounts*比*/etc/mtab*文件能更加真实的反映当前*mount*的情况。

### Linux命令的组成规则及其全拼单词

* su：Swith user 切换用户，切换到root用户
* cat: Concatenate 串联
* uname: Unix name 系统名称
* df: Disk free 空余硬盘
* du: Disk usage 硬盘使用率
* chown: Change owner 改变所有者
* chgrp: Change group 改变用户组
* ps：Process Status 进程状态
* tar：Tape archive 解压文件
* chmod: Change mode 改变模式
* umount: Unmount 卸载
* ldd：List dynamic dependencies 列出动态相依
* insmod：Install module 安装模块
* rmmod：Remove module 删除模块
* lsmod：List module 列表模块
* alias :Create your own name for a command
* bash :GNU Bourne-Again Shell linux内核
* grep:global regular expression print
* httpd :Start Apache
* ipcalc :Calculate IP information for a host
* ping :Send ICMP ECHO\_Request to network hosts
* reboot: Restart your computer
* sudo:Superuser do
* mnt=mount 英文解释： 登上; 爬上; 攀登; 骑上; 乘上; 跨上 可直接理解为“挂载”
* /bin = BINaries
* /dev = DEVices
* /etc = 存放配置文件的地方。配置文件的目录
  + - Editable Text Configuration 初期etcetra directory（ETCetera）,后来"Editable Text Configuration" 或者 "Extended Tool Chest"。
* /opt = Optional application software packages
* pwd = print working Directory (打印工作目录)。
* /lib = LIBrary
* /proc = PROCesses
* /sbin = Superuser BINaries
* /tmp = TeMPorary
* /usr = Unix Shared Resources
* /var = VARiable ? 是储存各种变化的文件，比如log等等
* FIFO = First In, First Out
* GRUB = GRand Unified Bootloader
* IFS = Internal Field Seperators
* LILO = LInux LOader
* MySQL = My最初作者的名字SQL = Structured Query Language
* PHP = Personal Home Page Tools = PHP Hypertext Preprocessor
* PS = Prompt String
* Perl = "Pratical Extraction and Report Language" = "Pathologically Eclectic Rubbish Lister"
* Python Monty Python's Flying Circus
* Tcl = Tool Command Language
* Tk = ToolKit
* VT = Video Terminal
* YaST = Yet Another Setup Tool
* apache = "a patchy" server
* apt = Advanced Packaging Tool
* ar = archiver
* as = assembler
* bash = Bourne Again SHell
* bc = Basic (Better) Calculator
* bg = BackGround
* cal = CALendar
* cat = CATenate
* cd = Change Directory
* chgrp = CHange GRouP
* chmod = CHange MODe
* chown = CHange OWNer
* chsh = CHange SHell
* cmp = compare
* cobra = Common Object Request Broker Architecture
* comm = common
* cp = CoPy
* cpio = CoPy In and Out
* cpp = C Pre Processor
* cups = Common Unix Printing System
* cvs = Current Version System
* daemon = Disk And Execution MONitor
* dc = Desk Calculator
* dd = Disk Dump
* df = Disk Free
* diff = DIFFerence
* dmesg = diagnostic message
* du = Disk Usage
* ed = editor
* egrep = Extended GREP
* elf = Extensible Linking Format
* elm = ELectronic Mail
* emacs = Editor MACroS
* eval = EVALuate
* ex = EXtended
* exec = EXECute
* fd = file descriptors
* fg = ForeGround
* fgrep = Fixed GREP
* fmt = format
* fsck = File System ChecK
* fstab = FileSystem TABle
* fvwm = F\*\*\* Virtual Window Manager
* gawk = GNU AWK
* gpg = GNU Privacy Guard
* groff = GNU troff
* hal = Hardware Abstraction Layer
* joe = Joe's Own Editor
* ksh = Korn SHell
* lame = Lame Ain't an MP3 Encoder
* lex = LEXical analyser
* lisp = LISt Processing = Lots of Irritating Superfluous Parentheses
* ln = LiNk
* lpr = Line PRint
* ls = list
* lsof = LiSt Open Files
* m4 = Macro processor Version 4
* man = MANual pages
* mawk = Mike Brennan's AWK
* mc = Midnight Commander
* mkfs = MaKe FileSystem
* mknod = MaKe NODe
* motd = Message of The Day
* mozilla = MOsaic GodZILLa
* mtab = Mount TABle
* mv = MoVe
* nano = Nano's ANOther editor
* nawk = New AWK
* nl = Number of Lines
* nm = names
* nohup = No HangUP
* nroff = New ROFF
* od = Octal Dump
* passwd = PASSWorD
* pg = pager
* pico = PIne's message COmposition editor
* pine = "Program for Internet News & Email" = "Pine is not Elm"
* ping = Packet InterNet Grouper
* pirntcap = PRINTer CAPability
* popd = POP Directory
* pr = pre
* printf = PRINT Formatted
* ps = Processes Status
* pty = pseudo tty
* pushd = PUSH Directory
* pwd = Print Working Directory
* rc = runcom = run command, shell
* rev = REVerse
* rm = ReMove
* rn = Read News
* roff = RunOFF
* rpm = RPM Package Manager = RedHat Package Manager
* rsh, rlogin, = Remote
* rxvt = ouR XVT
* sed = Stream EDitor
* seq = SEQuence
* shar = SHell ARchive
* slrn = S-Lang rn
* ssh = Secure SHell
* ssl = Secure Sockets Layer
* stty = Set TTY
* su = Substitute User
* svn = SubVersioN
* tar = Tape ARchive
* tcsh = TENEX C shell
* telnet = TEminaL over Network
* termcap = terminal capability
* terminfo = terminal information
* tr = traslate
* troff = Typesetter new ROFF
* tsort = Topological SORT
* tty = TeleTypewriter
* twm = Tom's Window Manager
* tz = TimeZone
* udev = Userspace DEV
* ulimit = User's LIMIT
* umask = User's MASK
* uniq = UNIQue
* vi = VIsual = Very Inconvenient
* vim = Vi IMproved
* wall = write all
* wc = Word Count
* wine = WINE Is Not an Emulator
* xargs = eXtended ARGuments
* xdm = X Display Manager
* xlfd = X Logical Font Description
* xmms = X Multimedia System
* xrdb = X Resources DataBase
* xwd = X Window Dump
* yacc = yet another compiler compiler

# 命令

## Ubuntu专区

## CentOS专区

## Manjaro专区

## 通用专区

### curl

如何在Curl中使用socks5代理？

|  |
| --- |
| curl -x socks5h://localhost:1080 http://www.google.com/ |

或

|  |
| --- |
| curl --socks5-hostname localhost:1080 http://www.google.com/ |

两个命令分别对应不同的版本，都试一下，哪个可用就用哪个。

# 常见错误

## Ubuntu专区

### The following schema is missing com.canonical.notify.osd

|  |
| --- |
| sudo apt-get install notify-osd |

## CentOS专区

## Manjaro专区

### 未知的公共密匙xxxx错误：一个或多个PGP签名无法校验

|  |
| --- |
| gpg --recv-keys <未知的密钥 xxxx> |

### 中文字体显示为方框

**原因：**当前使用的字体不支持中文或系统现有字体不支持中文汉字

**解决方案1：外观 > 字体**，更改为支持中文的字体

**解决方案2：**安装支持中文汉字的字体即可，可用pacman安装，也可以从网上下载ttf格式的字体复制到*/usr/share/fonts*下

## 通用专区

### gpg: symbol lookup error: gpg: undefined symbol: sqlite3\_errstr

问题原因：sqlite3版本太低，官网下载最新版sqlite3源码，升级即可。

问题原因相关讨论：<https://github.com/heroku/heroku-buildpack-python/issues/712>

sqlite3源码安装方法参看：sqlite3

# 附录