Windows 10 子系统 – Ubuntu

# 安装子系统

## 环境准备

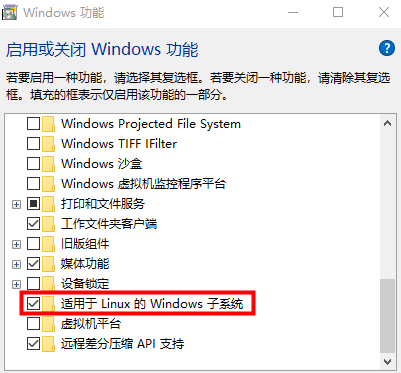
### 开启开发人员模式

Windows → 设置 → 更新和安全 → 开发者选项 → 使用开发人员功能 → 开发人员模式：选中，如下图所示：



### 开启子系统功能

控制面板 → 程序和功能 → 启用或关闭 Windows 功能 → 适用于 Linux 的 Windows 子系统：选中，如下图所示：



然后重启 Windows 系统，才能正常安装子系统。

## 安装过程

在线安装比较简单，但是会安装在C盘目录下，也就是安装在如下默认位置上：

C:\Users\Administrator\AppData\Local\Packages

这样会导致C盘越来越大。

建议采用离线安装方式，也比较简单，同时可以在任何位置安装。之后的所有操作均是指离线安装方式的操作。

### WSL 文档

<https://docs.microsoft.com/zh-cn/windows/wsl/>

### 在线下载

默认位置安装。

MicrosoftStore → 搜索 Linux → 找到 Ubuntu 或者 CentOS 进行下载即可，然后进行启动，即可进入安装过程。

### 离线下载

指定位置安装。

打开[手动下载适用于 Linux 的 Windows 子系统发行版包](https://docs.microsoft.com/zh-cn/windows/wsl/install-manual)网址，也就是：

<https://docs.microsoft.com/zh-cn/windows/wsl/install-manual>

然后选择自己想要安装的子系统发行版，下载下来的是一个**.appx**的文件，比如：

**Ubuntu\_2004.2020.424.0\_x64.appx**。我们把该文件复制到我们想要安装的位置上，比如：**D:\SubSystem\Ubuntu**，然后对该**.appx**文件进行**解压缩**，**解压缩**后的目录结构如下图所示：



然后运行ubuntu.exe文件（原文件名为ubuntu2004.exe），会自动进入安装过程。

### 安装过程

安装过程会自动进行，我们根据提示进行用户名和密码的设置即可，同时设置root密码。

## 启动子系统

系统安装完成之后，记得把**D:\SubSystem\Ubuntu**加入到系统变量的Path环境变量，如下图所示：



然后打开Cmder，输入**ubuntu**即可进入Ubuntu子系统。

## 重启子系统

如何在不重启Windows系统的情况下重启Linux子系统呢？

以管理员身份运行cmd命令提示符，然后执行如下命令：

|  |
| --- |
| Using CMD (Administrator) |
| net stop LxssManager  net start LxssManager |

## 子系统的重置/卸载/修复操作

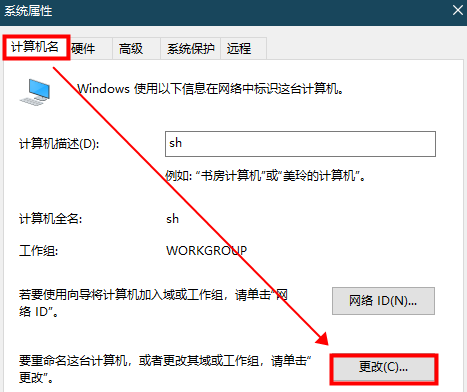
<https://jingyan.baidu.com/article/48a42057208777a924250496.html>

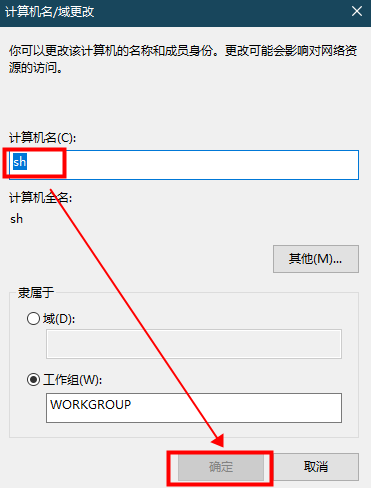
# 使用子系统

## 修改主机名

在Linux子系统的的主机名其实使用的就是宿主机Windows的计算机名，所以修改子系统额主机名其实就是通过修改Windows的计算机名来完成的。

右键“电脑” → 属性 → 计算机名、域和工作组设置 → 更改设置，如下图所示：





修改完之后，进行Windows系统重启即可。

## 文件共享

Linux子系统共享Windows中的文件：

执行如下命令：

|  |
| --- |
| cd /mnt/  ls |

这样就列出了**Windows系统中的盘符**，这样就可以进一步共享到Windows中的文件了。

简单来说就是 Windows 下的驱动盘已经被挂载到 Linux 子系统的 mnt 节点下了。

Windows共享Linux子系统中的文件：

这个也很简单，我们在Linux子系统中把文件复制到mnt节点下的某个盘符路径中即可。

## 修改镜像源

|  |
| --- |
| sudo vi /etc/apt/sources.list |
| 在该文件的最上面添加如下代码： |
| deb http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse  deb-src http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse  deb http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse  deb-src http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse  deb http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multiverse  deb-src http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multiverse  deb http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiverse  deb-src http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiverse  deb http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multiverse  deb-src http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multiverse |

然后保存即可。

## 系统更新

|  |
| --- |
| sudo apt-get update  sudo apt-get upgrade  sudo apt-get dist-upgrade |

## 软件安装

### 安装 Python3

插件快捷键指向：

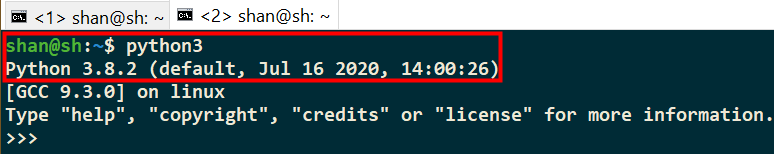
|  |
| --- |
| ls -l /usr/bin/ | grep python  ls -l /usr/bin/ | grep pip |

下载和安装 Python3

|  |
| --- |
| wget https://www.python.org/ftp/python/3.7.8/Python-3.7.8.tgz |
|  |
|  |
|  |

### 安装 pip3

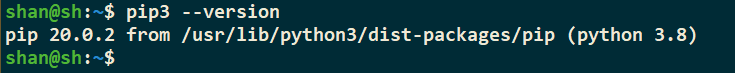
系统已经自带安装了Python3，如下图所示：



但是没有安装pip3，需要我们自己手工安装，安装命令如下：

|  |
| --- |
| sudo apt install python3-pip |

然后就可以使用pip3命令了，如下图验证结果：



### 安装 git

|  |
| --- |
| sudo apt-get install git |
| shan@sh:~$ git --version  git version 2.25.1 |

编译安装：

打开 <https://git-scm.com/download/linux> 查看第一段中的下载链接；

wget <https://www.kernel.org/pub/software/scm/git/git-2.28.0.tar.gz>

tar -zxvf git-2.28.0.tar.gz

cd git-2.28.0

sudo yum remove git

sudo yum -y install curl-devel expat-devel gettext-devel openssl-devel zlib-devel gcc perl-ExtUtils-MakeMaker

./configure --prefix=/usr/local/git

sudo make

sudo make install

sudo ln -s /usr/local/git/bin/\* /usr/bin/

[shan@sh ~]$ which git

/usr/bin/git

[shan@sh ~]$ git --version

git version 2.28.0

### bash shell 的美化

#### 安装 zsh shell

zsh 官网地址：

<https://www.zsh.org/>

<https://github.com/zsh-users/zsh>

|  |
| --- |
| sudo apt-get install zsh  zsh --version  zsh 5.8 (x86\_64-ubuntu-linux-gnu) |

刚好安装了最新版5.8版本。

查看安装结果：

|  |
| --- |
| shan@sh:~$ which zsh  /usr/bin/zsh |

加入shell名单：

|  |
| --- |
| nvim /etc/shells |



安装完成之后，在终端输入zsh命令，提示如下：

|  |
| --- |
| This is the Z Shell configuration function for new users,  zsh-newuser-install.  You are seeing this message because you have no zsh startup files  (the files .zshenv, .zprofile, .zshrc, .zlogin in the directory  ~). This function can help you with a few settings that should  make your use of the shell easier.  You can:  (q) Quit and do nothing. The function will be run again next time.  (0) Exit, creating the file ~/.zshrc containing just a comment.  That will prevent this function being run again.  (1) Continue to the main menu.  (2) Populate your ~/.zshrc with the configuration recommended  by the system administrator and exit (you will need to edit  the file by hand, if so desired).  --- Type one of the keys in parentheses --- |

按回车键，提示如下：

|  |
| --- |
| Aborting.  The function will be run again next time. To prevent this, execute:  touch ~/.zshrc |

意思是说，我们需要创建一下zsh的配置文件，否则上面的信息还会再次提醒。所以，我们执行如下命令创建配置文件：

|  |
| --- |
| sh% touch ~/.zshrc |

当再次进入zsh时，就没有上面的大段的提示信息了。

到这里zsh的安装过程就完成了。至于配置文件则留着上面创建的空文件即可。等安装完oh-my-zsh时，再添加配置即可。

卸载zsh：

|  |
| --- |
| sudo apt-get remove zsh |

#### 安装 oh-my-zsh

oh-my-zsh 官网地址：

<https://ohmyz.sh/>

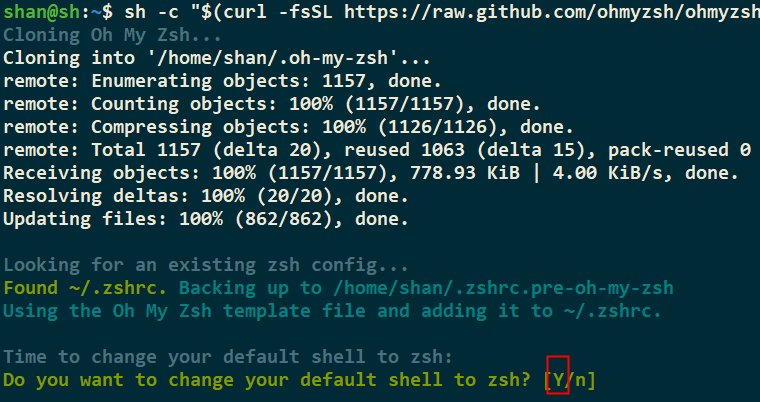
GitHub地址：<https://github.com/ohmyzsh/ohmyzsh>

安装文档：[https://ohmyz.sh/#install](https://ohmyz.sh/" \l "install)

执行如下安装命令：

|  |
| --- |
| sh -c "$(curl -fsSL https://raw.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh)" |
| 或者 |
| sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)" |

安装过程如下：



按Y回车，则把默认的shell修改为zsh。这时候，**重新打开.zshrc配置文件，里面已经不是空的了**。

卸载 oh-my-zsh：

|  |
| --- |
| cd ~/.oh-my-zsh/tools  chmod +x uninstall.sh  ./uninstall.sh  rm -rif .zshrc |

#### 配置 zsh shell

主题配置：

自带的主题所在的目录为：~/.oh-my-zsh/themes

把主题修改为：agnoster

|  |
| --- |
| ZSH\_THEME="robbyrussell" |
| 修改为 |
| ZSH\_THEME="agnoster" |
| 该主题使用说明：  <https://github.com/agnoster/agnoster-zsh-theme> |

使用该主题需要安装如下字体：

<https://github.com/powerline/fonts>

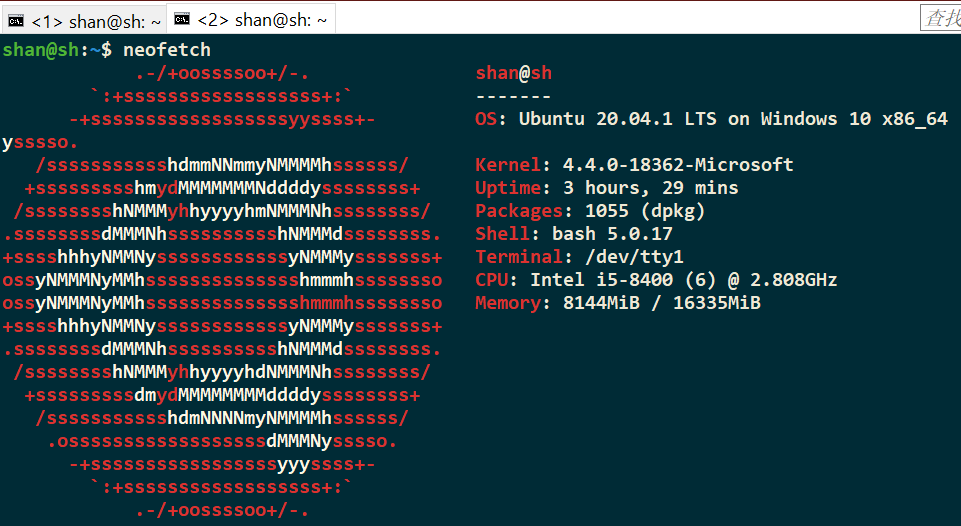
字体安装步骤：

|  |
| --- |
| git clone https://github.com/powerline/fonts.git --depth=1  cd fonts  ./install.sh  cd ..  rm -rf fonts  或者  sudo apt-get install fonts-powerline |

### 安装 neofetch 软件

|  |
| --- |
| sudo apt install neofetch |
| shan@sh:~$ neofetch |

如下效果图：



# 搭建开发环境

## 准备工作

我们将使用NeoVim做为开发工具。

使用vim-plug做为插件管理器。

使用coc.nvim做为代码补全插件。

### 安装 NeoVim 软件

NeoVim的官网地址为：<https://neovim.io/>

NeoVim的用户手册：<https://neovim.io/doc/user/>

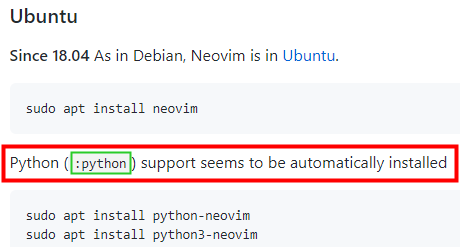
NeoVim的安装说明文档：

<https://github.com/neovim/neovim/wiki/Installing-Neovim>

因此Ubuntu的安装NeoVim的命令如下：

|  |
| --- |
| sudo apt-get install neovim |

在安装的过程中，会自动安装 python3-neovim，如下图说明所示：



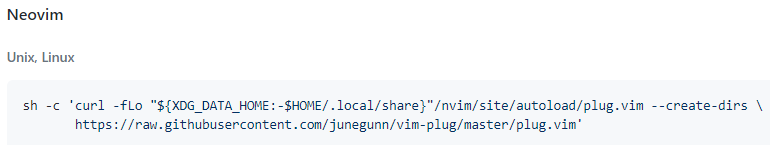
验证方法：打开NeoVim，然后在命令模式下输入:python发现确实存在，说明python3-neovim的确是安装了。

### 安装 vim-plug 插件管理器

vim-plug的官网地址为：

<https://github.com/junegunn/vim-plug>

安装命令如下图所示：

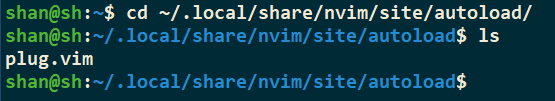


也就是：

|  |
| --- |
| sh -c 'curl -fLo "${XDG\_DATA\_HOME:-$HOME/.local/share}"/nvim/site/autoload/plug.vim --create-dirs https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim' |

我们在Ubuntu子系统的终端输入以上命令进行安装vim-plug插件管理器。

安装结果如下图所示：



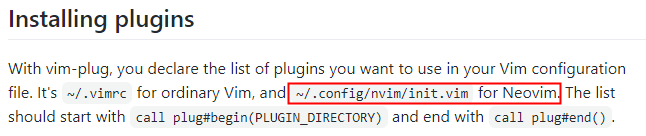
成功安装了vim-plug插件管理器。

### 启动 vim-plug 插件管理器

NeoVim如何启动 vim-plug 插件管理器呢？

打开网址：<https://github.com/junegunn/vim-plug/wiki/tutorial>

找到 “Installing plugins”，可以看到如下文字：



可以看到NeoVim配置文件的路径为：~/.config/nvim/init.vim。

我们根据需要创建相关的目录和文件：

|  |
| --- |
| shan@sh:~$ mkdir ~/.config  shan@sh:~$ mkdir ~/.config/nvim  shan@sh:~$ touch ~/.config/nvim/init.vim  shan@sh:~$ |

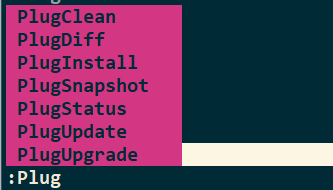
我们对init.vim文件进行编辑：

|  |
| --- |
| shan@sh:~$ nvim ~/.config/nvim/init.vim |

然后在文件中添加如下代码：

|  |
| --- |
| " 配置插件保存路径  call plug#begin('~/.config/nvim/plugged')  call plug#end() |

这样既启动了插件，同时又配置了插件的安装目录，我们启动NeoVim，在命令模式下查看是否有Plug操作的相关命令提示，如下图所示：



说明我们配置文件的位置是没问题的。

### 安装 coc.nvim 代码提示插件

coc.nvim的官网地址为：<https://github.com/neoclide/coc.nvim>

打开官网之后，可以看到安装coc.nvim插件之前，需要先安装nodejs软件，如下图所示：



因此，我们先安装nodejs、为了方便之后的操作，我们把npm和yarn也安装上。

命令如下所示：

|  |
| --- |
| sudo apt-get install nodejs npm yarn |

验证安装结果：

|  |
| --- |
| shan@sh:~$ node --version  v10.19.0 |
| shan@sh:~$ which npm  /usr/bin/npm  shan@sh:~$ npm --version  6.14.4 |
| shan@sh:~$ which yarn  /usr/bin/yarn  shan@sh:~$ yarn --version  0.32+git |

然后安装 coc.nvim 插件，我们编辑NeoVim的配置文件，添加如下代码：

|  |
| --- |
| Plug 'neoclide/coc.nvim', {'branch': 'release'} |

然后执行 :PlugInstall 进行插件的安装。

### 编写 NeoVim 配置文件

### 打包备份子系统

如何给子系统做备份呢？

以管理员身份打开cmd，然后执行如下命令：

|  |
| --- |
| net stop LxssManager |

然后给子系统做个备份（其实就是打个压缩包）。

## 搭建 前端 开发环境

|  |
| --- |
| :CocInstall coc-tsserver coc-json coc-html coc-css |

## 搭建 PHP 开发环境

## 搭建 Python 开发环境

系统自带已经安装了Python3.8版本，所以我们不需要再次安装Python了。

pip3 install -i https://pypi.doubanio.com/simple/ jedi-language-server

安装完成之后，提示说安装目录为 /home/shan/.local/bin/ 目录下。

:CocInstall coc-jedi

编辑 .py 文件时，提示如下错误信息：

[coc.nvim]: /bin/sh: 1: /home/shan/.config/coc/extensions/node\_modules/coc-jedi/.venv/bin/jedi-language-server: not found

进入 /home/shan/.config/coc/extensions/node\_modules/coc-jedi/.venv/bin 目录

创建软连接 ln -s /home/shan/.local/bin/jedi-language-server jedi-language-server

这样就没问题了。

## 搭建 Java 开发环境

1. 安装 Java 8 或以上版本的 JDK

输入: sudo apt install openjdk- 就会有安装版本提示，我们选择 openjdk-8 版本安装：

sudo apt install openjdk-8-jdk

验证安装是否成功:

shan@DESKTOP-NF0N71E:~$ which java

/usr/bin/java

shan@DESKTOP-NF0N71E:~$ java -version

openjdk version "1.8.0\_265"

OpenJDK Runtime Environment (build 1.8.0\_265-8u265-b01-0ubuntu2~20.04-b01)

OpenJDK 64-Bit Server VM (build 25.265-b01, mixed mode)

shan@DESKTOP-NF0N71E:~$

说明安装成功，且自动加入 PATH 环境变量。

2. :CocInstall coc-java

3. 编辑 .java 文件

通过编辑 .java 文件，自动下载 jdt.ls，如下提示信息：

[coc.nvim] jdt.ls not found, downloading...

touch Hello.java

nvim Hello.java

如何手工安装 jdt.ls 服务？

a. 打开 https://download.eclipse.org/jdtls/snapshots/?d 下载最新版本

https://download.eclipse.org/jdtls/snapshots/jdt-language-server-0.60.0-202008281606.tar.gz

这个压缩包下载将近 50 分钟，所以通过 NeoVim 自动安装就更加慢了。

b. 把压缩包解压缩到 ~/.config/coc/extensions/coc-java-data/server/ 目录下

在解压缩之前，先清空 server 目录下的所有文件和目录

rm -rf \*

tar -zxvf jdt-language-server-0.60.0-202008281606.tar.gz

c. :CocCommand java.updateLanguageServer

[coc.nvim] Command: java.updateLanguageServer not found

:h coc-status → 检查状态

[coc.nvim] The "java" server crashed 5 times in the last 3 minutes. The server will not be restarted.

java.jdt.ls.vmargs : "-noverify -Xmx1G -XX:+UseG1GC -XX:+UseStringDeduplication"

手工安装 jdt.ls 服务的参考网址:

https://github.com/neoclide/coc-java/issues/12

插件安装参考网址:

https://github.com/neoclide/coc-java

## 搭建 Go 开发环境

1.安装和配置 go 环境

wget https://golang.org/dl/go1.15.linux-amd64.tar.gz

tar -zxvf go1.15.linux-amd64.tar.gz

安装参考: https://golang.org/doc/install

以下是详细安装步骤:

cd /usr/local/

wget https://golang.org/dl/go1.15.linux-amd64.tar.gz

[sudo cp /mnt/d/ShareToLinux/go1.15.linux-amd64.tar.gz /usr/local/]

sudo tar -zxvf go1.15.linux-amd64.tar.gz

在 ~/.profile 添加如下语句:

export PATH=$PATH:/usr/local/go/bin/

source ~/.profile

2.安装 gopls 模块

echo "export GOPROXY=https://goproxy.cn" >> ~/.profile && source ~/.profile

go get golang.org/x/tools/gopls@latest

出现如下错误:

go: cannot use path@version syntax in GOPATH mode

echo "export GO111MODULE=on" >> ~/.profile && source ~/.profile

3.安装和配置插件

https://github.com/fatih/vim-go

https://github.com/golang/tools/tree/master/gopls

https://github.com/golang/tools/blob/master/gopls/doc/vim.md

把以下代码写入 NeoVim 配置文件

```shell

Plug 'fatih/vim-go', {'do': ':GoUpdateBinaries'}

let g:go\_def\_mode='gopls'

let g:go\_info\_mode='gopls'

```

然后执行插件安装命令。

4.安装代码提示插件

然后执行 :CocInstall coc-go 进行插件安装，然后就可以有 go 的代码提示了。

5. 出现 go.mod 错误

在编辑 .go 文件时，出现如下错误提示信息:

go: cannot find main module; see 'go help modules'

这时，我们需要在项目根目录下面创建 go.mod 文件，创建过程如下:

```

go mod init [module] → 这里的 [module] 替换为项目名称

```

即可。

出现 go.mod 错误的参考网址: https://blog.csdn.net/benben\_2015/article/details/82227338

6. 可以开始愉快的使用 Go 开发了。

## 搭建 C/C++ 开发环境

## 搭建 C# 开发环境