

目录

简介	1.1
ESP32 IoT 开发指南 总纲	1.2
一、 ESP32 开发环境搭建	1.3
WSL 简明使用教程 — 安装&配置	1.3.1
WSL + VS Code 搭建 ESP32 开发环境	1.3.2
ESP-IDF 安装器 + VS Code 搭建 ESP32 开发环境	1.3.3
二、 ESP32 IOT 开发框架与编译系统	1.4
ESP-IDF SDK 框架解析	1.4.1
ESP-IDF SDK 结构 - 组件(components)	1.4.1.1
ESP-IDF SDK 结构 - Kconfig(components)	1.4.1.2
ESP-IDF 构建系统	1.4.2
GUN Make 构建系统	1.4.2.1
CMake 构建系统	1.4.2.2
三、 ESP32 基础开发指南	1.5
ESP32 常用外设开发	1.5.1
ESP32 GPIO 操作	1.5.1.1
ESP32 UART 操作	1.5.1.2
ESP32 IIC 操作	1.5.1.3
ESP32 SPI 操作	1.5.1.4
ESP32 WIFI & Internet 开发	1.5.2
ESP32 连接 WIFI AP(Station 模式)	1.5.2.1
ESP32 创建 Soft AP(Soft AP 模式)	1.5.2.2
ESP32 Station + Soft AP共存	1.5.2.3
ESP32 BLE 例程	1.5.3
Bluetooth &&BLE 基础知识扫盲	1.5.3.1
BLE 协议栈 - 物理层(Physical Layout)	1.5.3.1.1
BLE 协议栈 - 链路层(Link Layout)	1.5.3.1.2
BLE 协议栈 - HCI(Host Controller Interface)	1.5.3.1.3
BLE 协议栈 - L2CAP(Logical Link Control and Adaptation Protocol)	1.5.3.1.4
BLE 协议栈 - ATT(Attribute Protocol)	1.5.3.1.5
BLE 协议栈 - GATT(General Agreement on Tariffs and Trade)	1.5.3.1.6
BLE 协议栈 - GAP(Generic Access Profile)	1.5.3.1.7
四、 ESP32 进阶开发指南	1.6
ESP32 功能组件开发	1.6.1
ESP32 控制台组件 - Console	1.6.1.1
ESP32 文本处理组件 - cJSON	1.6.1.2
ESP32 ESP-IDF 协议开发	1.6.2

MQTT 协议开发	1.6.2.1
MQTT 基础知识	1.6.2.1.1
HTTP/HTTPS 协议开发	1.6.2.2
HTTP 基础知识 - TCP 连接三次握手,断开四次握手	1.6.2.2.1
HTTP 基础知识 - GET/POST	1.6.2.2.2
Websocket 协议开发	1.6.2.3
ESP32 对接云端开发	1.6.3
对接 阿里智能云平台	1.6.3.1
对接 腾讯 IoT Explorer	1.6.3.2
对接 华为 HiLink	1.6.3.3
对接 京东小京鱼	1.6.3.4
对接 百度天工(百度物平台)	1.6.3.5
对接 Google IoT Core	1.6.3.6
对接 AWS-IoT	1.6.3.7
对接 AZURE	1.6.3.8
ESP32 ESP-IDF 框架分析	1.6.4
ESP32 ESP-IDF EVENT 框架分析	1.6.4.1
五、ESP32 高级开发指南(待更新)	2.1
ESP32 应用例程(待更新)	2.1.1
六、AURORA 蓝牙网关(更新中)	2.2
AURORA ESP32 SDK 简介	2.2.1
AURORA ESP32 SDK 框架/组件分析	2.2.2
七、开发工具及使用技巧	2.3
开发工具 - VS Code	2.3.1
开发工具 - 常用 shell 命令/脚本	2.3.2
开发工具 - make 命令/脚本	2.3.3
开发工具 - git 命令/脚本	2.3.4
开发工具 - python 命令/脚本	2.3.5
开发工具 - Kconfig 使用	2.3.6
开发工具 - Postman 使用	2.3.7
开发调试/编辑软件/工具 - 推荐	2.3.8
结束	2.4

- [ESP32 IoT 开发指南](#)



ESP32 IoT 开发指南

本系列文章由 [ET Team@QinYUN575](#) 为 ESP32 在物联网的实际应用开发而撰写

优秀的人总是有共同之处——努力，孜孜不倦的往自己的方向前行。

“不积跬步无以至千里”，共励共勉！

- 如果本系列文章对你有帮助，请动动小手分享给你的朋友，谢谢！

——码于三月三号二零一九年

更新时间 V0.2@20191128

```
**[terminal]
**[prompt foo@joe]**[path ~]**[delimiter $]**[command ./myscript]
Normal output line. Nothing special here...
But...
You can add some colors. What about a warning message?
**[warning [WARNING] The color depends on the theme. Could look normal too]
What about an error message?
**[error [ERROR] This is not the error you are looking for]
```

-
- [GitHub@QinYUN575](#)
 - [Blog@柯钿爽的个人博客](#)
-

ESP32 IoT Guide by ET Team@QinYUN575

Copyright © qinun575@foxmail.com 2019 all right reserved , powered by Gitbook 该文件修订时间：2019-11-29 08:57:00

- 前言
- 一、关于 ESP32
 - 摘要
 - ESP32 SOC 片上资源
 - 开发框架/平台资源
 - ESP32 开发教程
- 二、ESP32 开发环境搭建&开发框架指南
 - 简介
 - ESP32 开发环境搭建
 - ESP32 IOT 开发框架与编译系统
- 三、ESP32 基础开发指南
 - ESP32 外设例程
 - ESP32 WIFI & INTERNET 例程
 - ESP32 BLE 例程
 - ESP32 FreeRTOS 例程
- 四、ESP32 进阶开发指南
 - ESP32 SIG MESH 例程
 - ESP32 ADF 例程
 - ESP32 MDF 例程
- 五、ESP32 高级开发指南
 - ESP32 应用例程
- 六、ESP32 Geek 开发指南
 - ESP32 MicroPython
 - ESP32 Ardunio
 - ESP32 Javascript
 - ESP32 Geek 设计
 - ESP32 HackCube 设计

前言

这篇文章是 "ESP32 开发指南(非乐鑫官方)" 教程的总纲；
对于很多从事嵌入式开发的朋友应该都对 ESP8266 非常熟悉了！在 ESP8266 大热之后，乐鑫继而在 2015 年推出功能强劲的升级版 WIFI SOC: **ESP32**；
然而我在学习/开发 ESP32 过程中却遇到很多坑，网上的资料现在零零碎碎的，故而写下本系列文章，希望能对在学习/开发 ESP32 践坑中的你，能有少许帮助，谢谢！

——码于三月三号二零一九年

备注:

本系列文章基于 [Espressif esp-idf Realse V4.x 版本](#)

核心开发板 by [安信可 Ai-thinker@NodeMCU-32S](#)



安信可科技
Ai-Thinker

ISO9001认证



一、关于 ESP32

摘要

ESP32 是今年科创板上的新星公司——乐鑫信息科技有限公司于 2015 年推出的 WiFi SOC;ESP32 芯片集成 802.11b/g/n/i WiFi 和 Bluetooth 4.2 RF，并搭载两个 32 位 Tensilica LX6 MCU，最高主频可达 600DMIPS，可直接传输音频流、视频流等，功能非常之强劲。

Wi-Fi 和蓝牙解决方案

ESP32 可作为独立系统运行应用程序或是主机 MCU 的从设备，通过 SPI / SDIO 或 I2C / UART 接口提供 Wi-Fi 和蓝牙功能。

超低功耗

该款芯片拥有业内最高水平的低功耗芯片的所有特征，比如精细分辨时钟门控、省电模式和动态电压调整等。

性能稳定

ESP32 集成了更多的元器件，性能稳定，易于制造，工作温度范围达到 -40°C 到 $+125^{\circ}\text{C}$ 。ESP32 还集成了先进的自校准电路，实现了动态自动调整，可以消除外部电路的缺陷以及适应外部条件的变化。

ESP32 SOC 片上资源

ESP32 基本参数		
类型	项目	规格
CPU SOC	CPU	ESP32, 内置两个低功耗 Xtensa® 32-bit LX6 MCU(80Mhz/160Mhz/240Mhz) 计算能力高达 650 DMIPS 一个超低功耗协处理器 ULP
	ROM	448 KB 的 ROM
	SRAM	用于数据和指令存储的 520 KB 片上 SRAM RTC 快速存储器，为 8 KB 的 SRAM RTC 慢速存储器，为 8 KB 的 SRAM
	eFuse 安全加密	1 Kbit 的 eFuse
Wi-Fi	协议	802.11 b/g/n (802.11n , 速度 150 Mbps) A-MPDU 和 A-MSDU 聚合，支持 0.4 μs 保护间隔
	频率范围	2.4 GHz ~ 2.5 GHz
蓝牙	协议	符合蓝牙 v4.2 BR/EDR 和 BLE 标准
	射频	具有 -97 dBm 灵敏度的 NZIF 接收器 Class-1, Class-2 和 Class-3 发射器 AFH
	音频	CVSD 和 SBC 音频
		<ul style="list-style-type: none"> • 34 个 GPIO 口 • 12-bit SAR ADC，多达 18 个通道 • 2 个 8-bit D/A 转换器

硬件	模组接口： 34 个 GPIO 口 模组 GPIO6 至 GPIO11 用于连接模组上集成的 SPI flash/PSRAM，不建议用于其他功能（可用 28 个）	<ul style="list-style-type: none"> • 10 个触摸传感器 • 4 个 SPI [可重映射] • 2 个 I_SS [可重映射] • 2 个 I_CC [可重映射] • 3 个 UART [可重映射] • 1 个 Host SD/eMMC/SDIO • 1 个 Slave SDIO/SPI • 带有专用 DMA 的以太网 MAC 接口，支持 IEEE 1588 • CAN2.0 • IR (TX/RX) • 电机 PWM • LED PWM，多达 16 个通道 • 霍尔传感器
	片上传感器	霍尔传感器
	工作电压/供电电压	2.7 V ~ 3.6 V
	最小供电电流	500 mA
	建议工作温度范围	-40 °C ~ 65 °C

more...

开发框架/平台资源

开发框架/解决方案	链接	说明
ESP-IDF	espressif@esp-idf	IoT 开发框架
ESP-ADF	espressif@esp-adf	音频开发框架
ESP-MDF	espressif@esp-mdf	WIFI-MESH 开发框架
esp-iot-stution	espressif@esp-iot-stution	IoT 解决方案
esp-jumpstart	espressif@esp-jumpstart	快速上手示例

more ...

ESP32 开发教程

乐鑫官方编程指南: [ESP32-IDF 编程指南](#)

二、ESP32 开发环境搭建&开发框架指南

简介

由于 ESP8266/ESP32 是乐鑫科技基于 Xtensa 内核定制的 WIFI SOC, 非 STM32,MM32 等主流 Cortex-M 内核架构，所以无法使用 MDK(Keil)/IAR 等主流 MCU 开发环境；而乐鑫主推的官方开发环境为 Linux, 但国内大部分嵌入式/单片机开发者都是在 Windows 上进行开发,所以我们下面的将在 Windows 上搭建我们的开发环境.

ESP32 开发环境搭建

开发环境介绍:

这里根据乐鑫官方的教程，以实际的开发情况，我们可以有以下几种开发环境供我们选择(GNU Make 逐渐弃用,换用 CMake):

			备
--	--	--	---

开发环境	说明	文章	注
WSL(Linux)+ Make + VS Code	WSL(Linux) + VS Code 搭建 ESP32 开发环境	该环境搭建分成两部分， 第一部分是安装配置 WSL: 《WSL 简明使用教程 — 安装&配置》 第二部分是在 WSL +VS Code 配置 ESP32 开发环境: 《WSL + VS Code 搭建 ESP32 开发环境》	更新中
CMake(Windows) + VS Code	使用 ESP-IDF 安装器 + VS Code 搭建 ESP32 开发环境	《ESP-IDF 安装器 + VS Code 搭建 ESP32 开发环境》	更新中
MSYS2(Windows) + VS Code	Win 7/8/10 + Msys2 + VS Code 搭建 ESP32 开发环境 (编译工程过慢,且易出错,官方弃用中)	-	待更新
-	-	-	-

WSL(Linux) + VS Code 搭建 ESP32 开发环境

WSL 即 "Windows Subsystem for Linux" 的缩写 , 顾名思义 , WSL 就是 Windows 系统下的 Linux 子系统 ,

WSL 作为 Windows 的组件搭载在 Windows 10 周年更新(1607) 之后的系统中,所以下面文章仅适用于 Windows 10 (1607) 之后的系统

该环境搭建分成两部分:

第一部分是 安装配置 WSL , 第二部分是在 WSL +VS Code 配置 ESP32 开发环境 , 所以分成两篇文章:

由于 WSL 是 Linux 系统, 故与 Linux 下搭建 ESP32 方法相同 Linux (WSL)系统使用 Ubuntu 18 LTS

一、 [《WSL 简明使用教程 — 安装&配置》](#)

二、 [《WSL + VS Code 搭建 ESP32 开发环境》](#)

使用 ESP-IDF 安装器 + VS Code 搭建 ESP32 开发环境

由于大部分嵌入式/MCU 开发工程师都是在 Windows 进行开发的,所以乐鑫推出了适用于 Windows 上搭建 ESP32 开发环境的安装器 ESP-IDF 工具安装器 (ESP-IDF-tools-setup)

下载地址: <https://dl.espressif.com/dl/esp-idf-tools-setup-2.0.exe>

我们可以使用 ESP-IDF 安装器 + VS Code 搭建 ESP32 开发环境 来快速搭建 ESP32 开发环境

[《ESP-IDF 安装器 + VS Code 搭建 ESP32 开发环境》](#)

Msyz2(Windows) + VS Code 搭建 ESP32 开发环境

- 暂不更新 (编译工程过慢,且易出错,官方弃用中)

ESP32 IOT 开发框架与编译系统

序号	笔记 & 教程	工程源码	备注
D000	D000.make	ESP32 构建系统之 make 指令 && makefile	None.
D001	D001.kconfig	ESP32 构建系统之 Kconfig	None.
D002	D002.menuconfig	menuconfig	None.

more ...

三、ESP32 基础开发指南

ESP32 外设例程

序号	笔记 & 教程	工程源码	备注
P001	P001 gpio_output_led	GPIO 点亮LED	None.
P0**	*	P0*_	*

more...

ESP32 WIFI & INTERNET 例程

序号	笔记 & 教程	工程源码	备注
W000	W000 SoftAP+STA	ESP32 Soft+AP&STA 模式使用	None.
W0**	*	W0*_	*

more...

ESP32 BLE 例程

序号	笔记 & 教程	工程源码	备注
B0**	*	B0*_	*

soon in...

ESP32 FreeRTOS 例程

序号	笔记 & 教程	工程源码	备注
F001	ESP32FreeRTOS任务管理	*	*
F002	ESP32FreeRTOS软件定时器	*	*
F003	ESP32FreeRTOS事件标志组	*	*
F004	ESP32FreeRTOS信号量-二值信号量	*	*
F005	ESP32FreeRTOS信号量-优先级翻转	*	*
F006	ESP32FreeRTOS信号量-互斥信号量	*	*
F007	ESP32FreeRTOS消息队列	*	*
F008	ESP32FreeRTOS任务通知	*	*
F0**	*	*	*

more...

四、ESP32 进阶开发指南

ESP32 SIG MESH 例程

序号	笔记 & 教程	工程源码	备注
MB0**	*	*	*

soon in...

ESP32 ADF 例程

序号	笔记 & 教程	工程源码	备注
A0**	*	*	*

soon in...

ESP32 MDF 例程

序号	笔记 & 教程	工程源码	备注
MW0**	*	*	*

soon in...

五、ESP32 高级开发指南

ESP32 应用例程

序号	笔记 & 教程	工程源码	备注
C0**	*	*	*

soon in...

六、ESP32 Geek 开发指南

ESP32 MicroPython

序号	笔记 & 教程	工程源码	备注
PY0**	*	*	*

soon in...

ESP32 Arduunio

序号	笔记 & 教程	工程源码	备注
AD0**	*	*	*

soon in...

ESP32 Javascript

序号	笔记 & 教程	工程源码	备注
JS0**	*	*	*

soon in...

ESP32 Geek 设计

序号	笔记 & 教程	工程源码	备注
G0**	*	*	*

soon in...

ESP32 HackCube 设计

序号	笔记 & 教程	工程源码	备注
H0**	*	*	*

soon in...

Copyright © qinun575@foxmail.com 2019 all right reserved , powered by Gitbook 该文件修订时间 : 2019-11-29 08:57:00

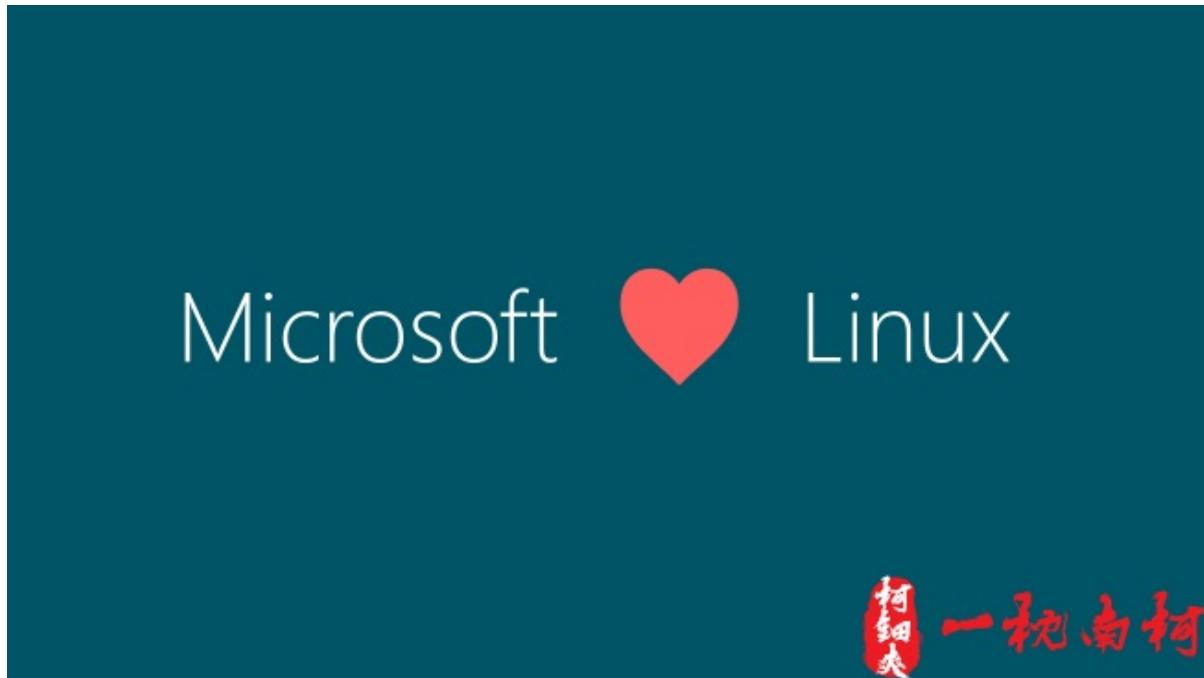
- 一、前言
 - 1.什么是 WSL
 - 2.WSL 能做什么
 - 3.前期准备
- 二、安装 WSL
 - 1.Windows 10 开启开发者模式 & 打开 Linux 子系统功能
 - 2.使用 Microsoft 应用商城 安装 WSL
- 三、对 WSL 进行一些基本配置
 - 1.切换软件源为 阿里源/清华源
 - 2.缩短显示路径
- 4.VIM 编辑器默认显示行号
- 4.安装 screefetch 显示系统信息(装X工具)
- 四、在 Visual Studio Code 中使用 WSL 做为默认终端

一、前言

摘要:

在本教程中,我们将简短的了解什么是 WSL, WSL 的安装和一些常规的配置,话不多说,那我们开始吧!

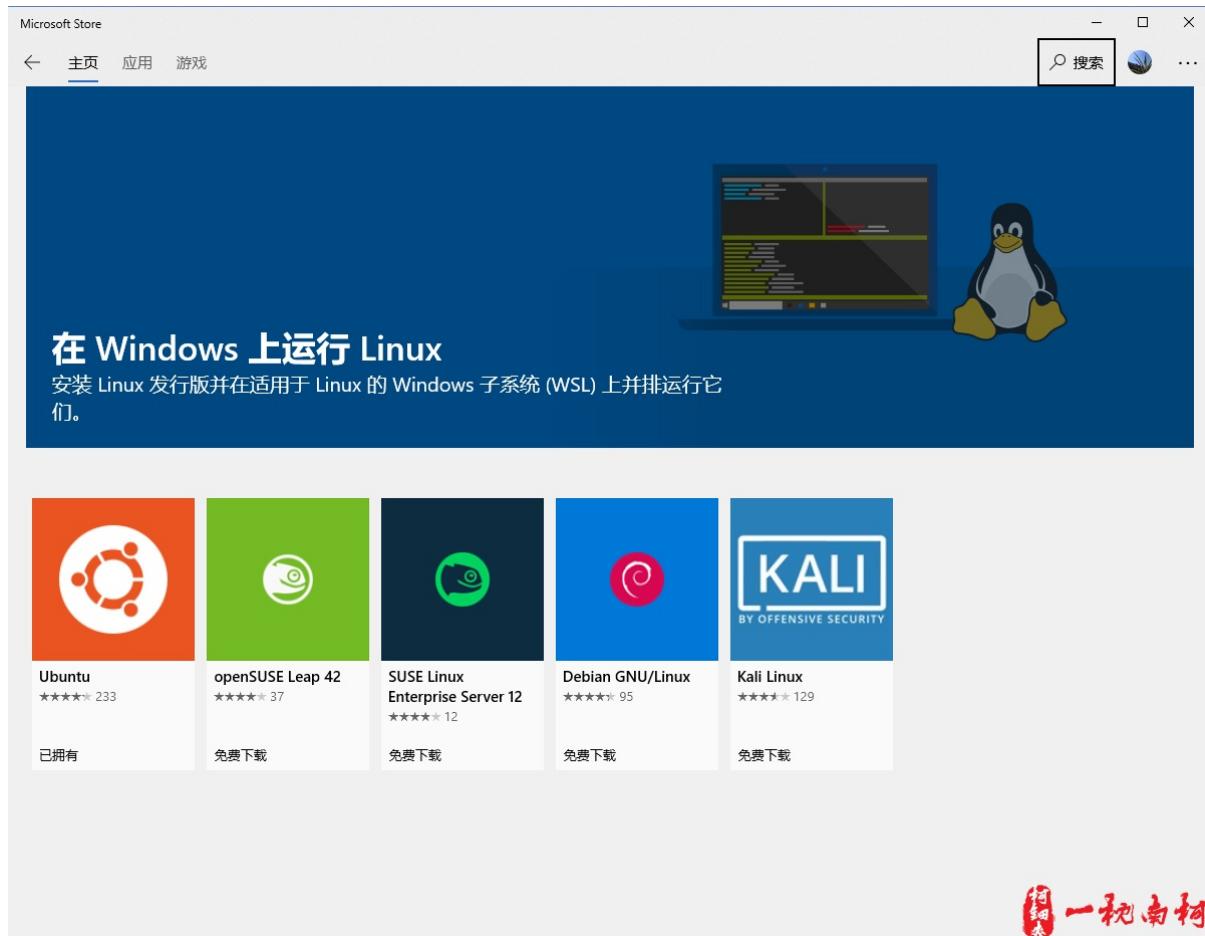
关键词: `WSL` `Windows 10` `Bash` `Ubuntu` `Linux`



1.什么是 WSL

WSL 即 "Windows Subsystem for Linux" 的缩写,顾名思义, wsl 就是 Windows 系统下的 Linux 子系统,WSL 作为 Windows 的组件搭载在 Windows 10 周年更新(1607)之后的系统中.

WSL 刚出来的时候只有 Ubuntu 16 这个发行版可以使用,而现在已经有很多 Linux 子系统发行版供我们使用了.



对于许多国内的开发者来说,Windows 才是主要的开发环境,但对于一些想要尝试 Linux 的开发者来说,经常需要因为一些 Windows 独享的开发软件,而不得不在 Windows 和 Linux 之间来回切换,实在麻烦. 现在有了 WSL ,在 Windows 下我们也可以尝试 Linux 的魅力,又可以抛弃难用到恶心的 CMD 了.

2.WSL 能做什么

既然是 Linux 子系统,那便是可以和 Linux 一样,当然由于是 Windows 系统,故一些关于硬件的驱动的相关驱动便没有那么完善,详情请自己 Google.

3.前期准备

- Windows 10 (版本号>1607)
- Microsoft 账号用于登陆 Microsoft Store
- 耐心

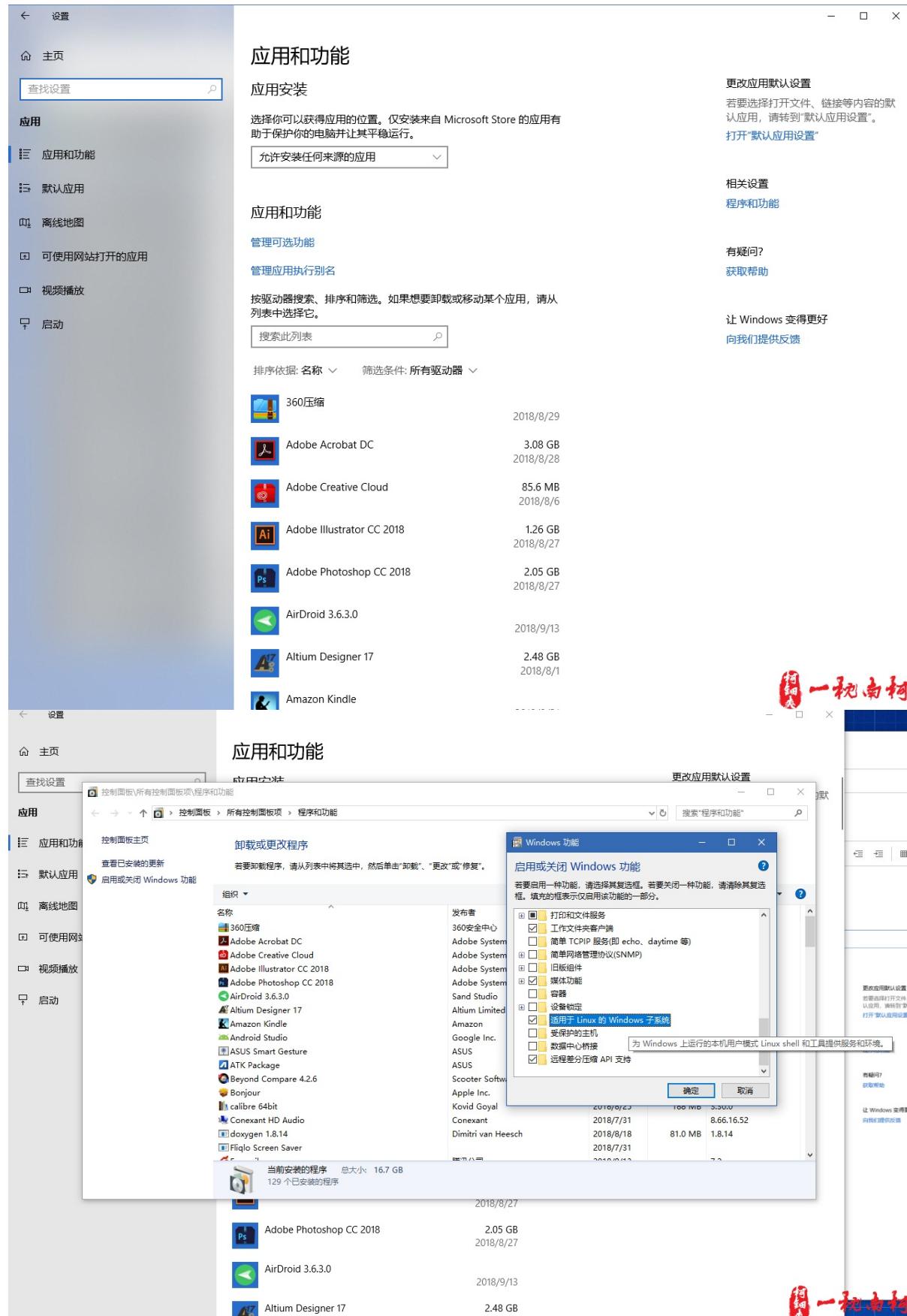
二、安装 WSL

1.Windows 10 开启开发者模式 & 打开 Linux 子系统功能

1.设置界面打开开发者模式



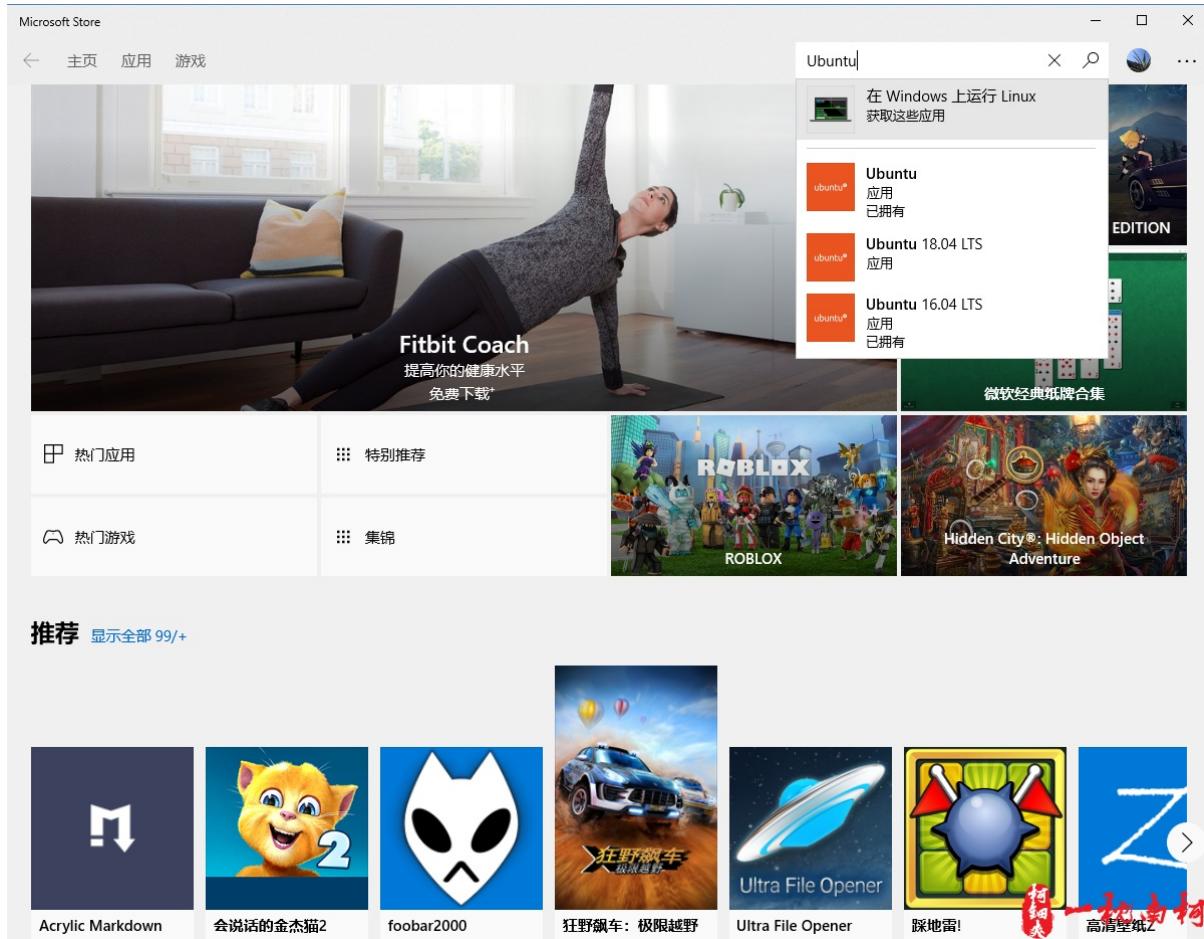
2.程序和功能->启用或关闭 Windows 功能 -> 勾选适用于 Linux 的 Windows 子系统

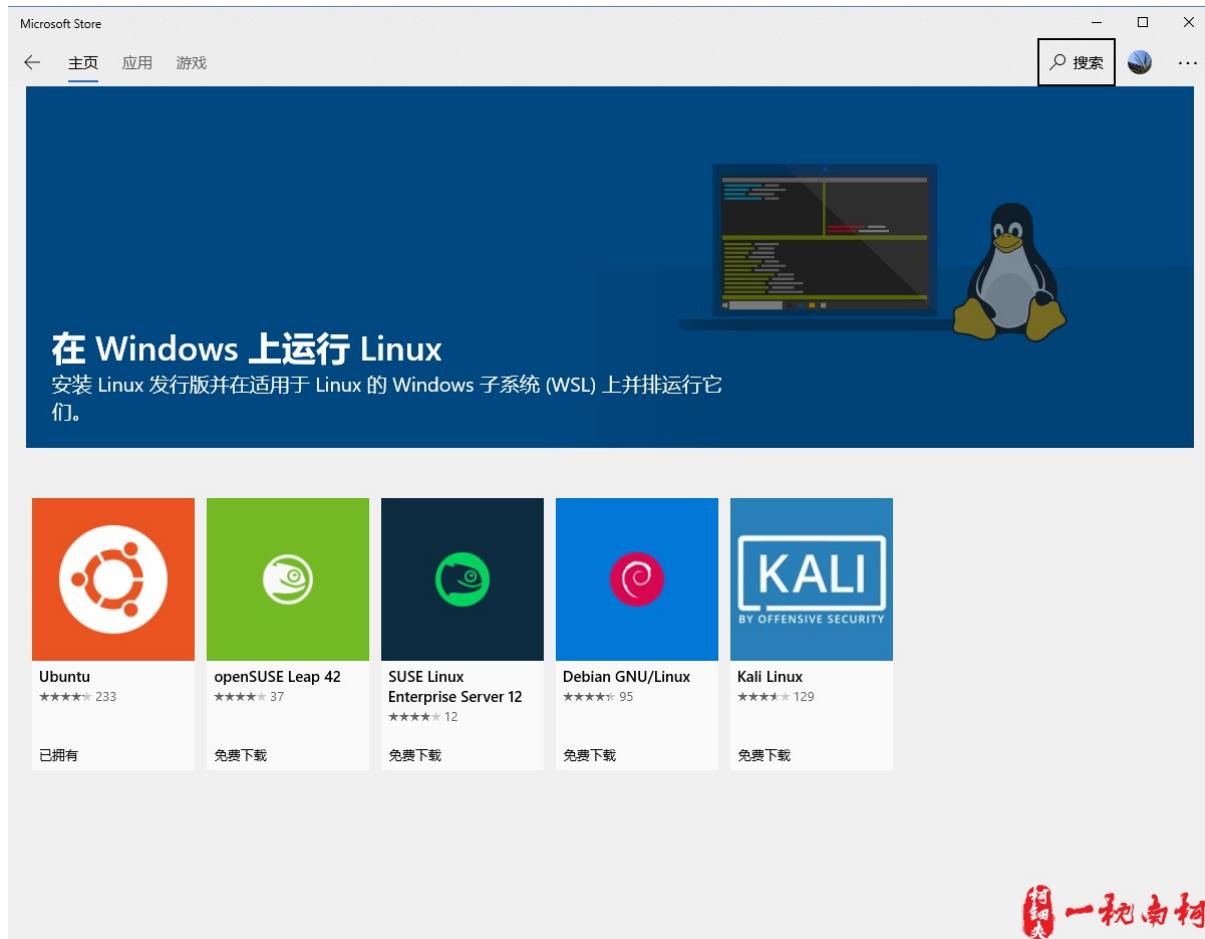


3.重启电脑

2. 使用 Microsoft 应用商城 安装 WSL

1. 打开并用微软账号登陆 Microsoft Store, 搜索安装 Linux 子系统发行版本: Ubuntu (Ubuntu 18)





一枕南柯

Ubuntu
Canonical Group Limited • ★★★★☆ 撰写评价
你拥有此产品。
安装 ...

用户也喜欢 显示全部

Termius - SSH client	Debian GNU/Linux	openSUSE Leap 42	X SERVER 4 WINDOWS 10	Ubuntu 18.04 LTS	HeidiSQL	Ubuntu 16.04 LTS	SUSE Linux Enterprise Server...
★★★★★ 5 免费下载	★★★★★ 95 免费下载	★★★★★ 37 免费下载	节省 ¥236.00 X410 ★★★★★ 5 ¥269.00 ¥33.00	★★★★★ 109 免费下载	★★★★★ 22 免费下载	★★★★★ 20 已拥有	★★★★★ 12 免费下载

描述

Ubuntu on Windows allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

Note that Windows 10 S does not support running this app.

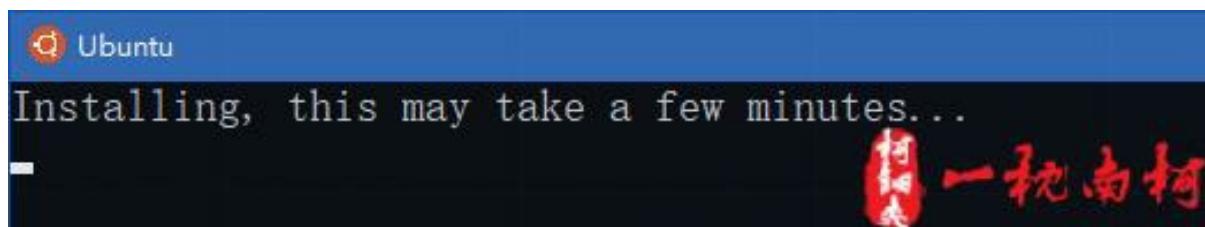
Before installing Ubuntu on Windows or before the first run please open the Control Panel, visit Programs and Features' submenu Turn Windows features on or off and select Windows Subsystem for Linux. Click OK, reboot, and then your

设备支持

电脑

2.启动 WSL,配置 linux 用户账户名称与密码

1.初始化 WSL 需要大概 5 分钟,耐心等待



2.输入 Linux 用户 名称&密码 (密码输入时不可见,输入两次确认)

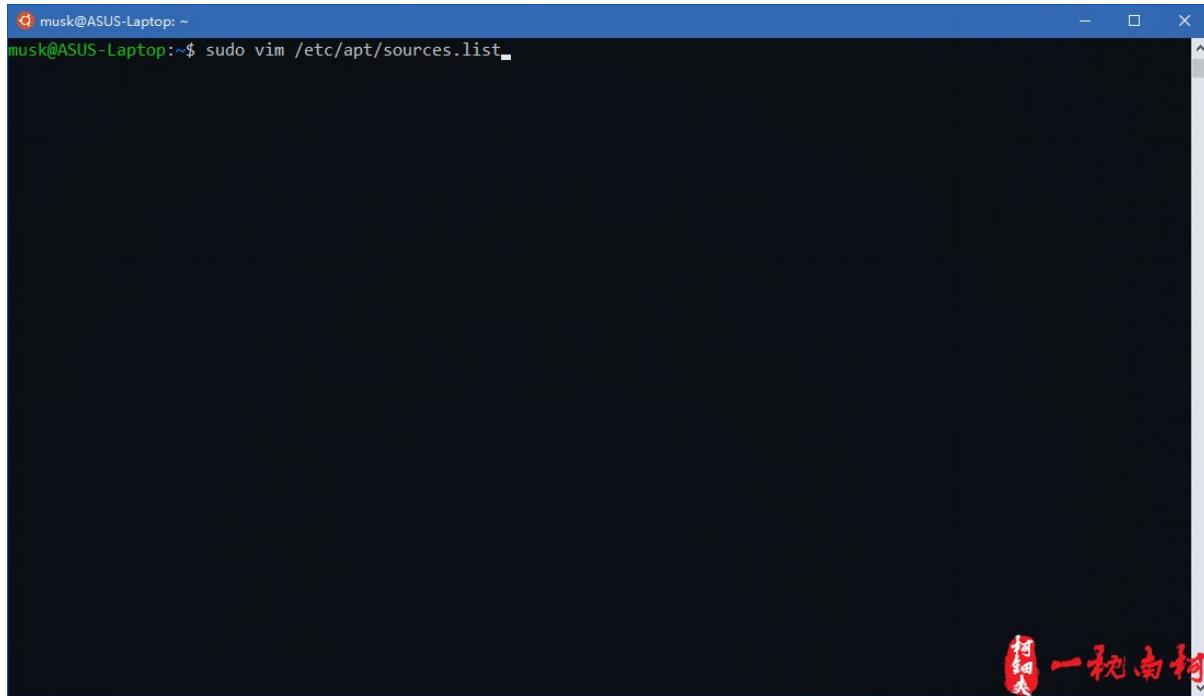
三、对 WSL 进行一些基本配置

1.切换软件源为 阿里源/清华源

1.在 WSL 中输入

```
**[terminal]
**[command sudo vim /etc/apt/sources.list]
```

输入 :%d ,删除所有内容,之后按 i 键进入 插入模式 ,

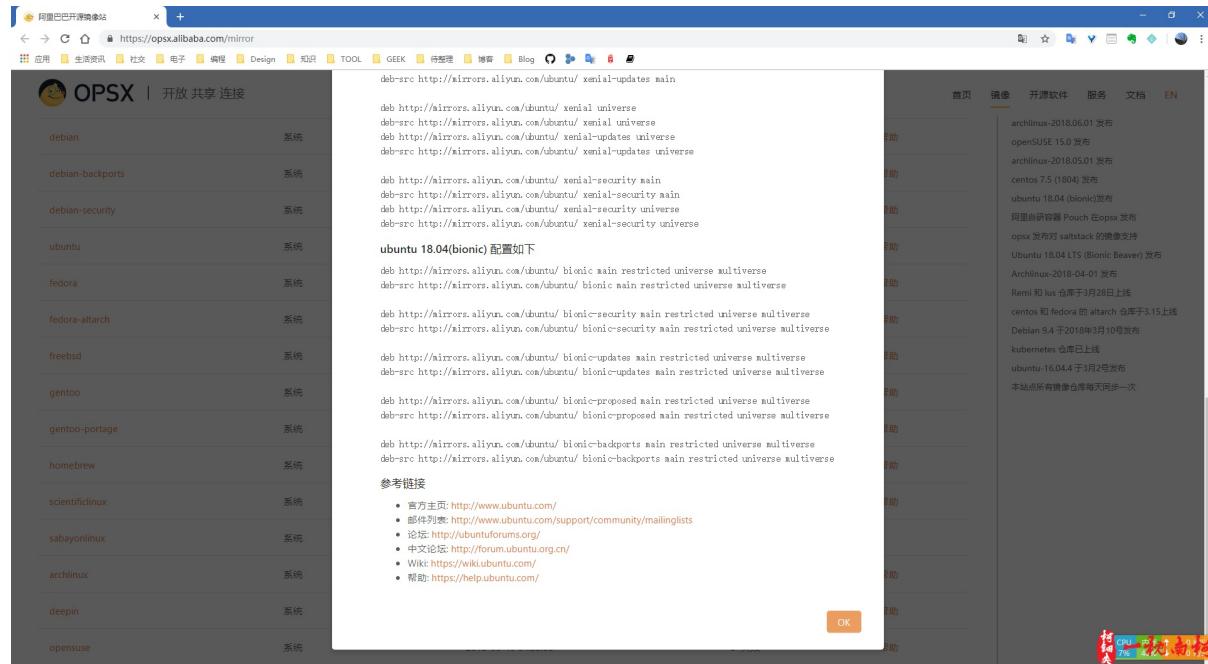


2. 打开阿里源 (阿里巴巴开源镜像站 <https://opsx.alibaba.com/mirror>)

The screenshot shows the OPSX website at <https://opsx.alibaba.com/mirror>. The page displays a table of mirrors for various Linux distributions, including Debian, Ubuntu, Fedora, and ArchLinux. The 'ubuntu' mirror is highlighted with a yellow background. The right sidebar contains a list of other mirrors and some system status information.

Distribution	Type	Last Update	Status	Action
debian	系统	2018-09-19 06:35:45	失败	帮助
debian-backports	系统	2018-09-19 01:30:11	成功	帮助
debian-security	系统	2018-09-19 01:36:54	成功	帮助
ubuntu	系统	2018-09-19 01:55:47	成功	帮助
fedora	系统	2018-09-19 05:46:30	成功	帮助
fedora-altarch	系统	2018-09-19 07:41:47	成功	帮助
freebsd	系统	2018-09-19 02:54:02	成功	帮助
gentoo	系统	2018-09-19 02:01:21	成功	帮助
gentoo-portage	系统	2018-09-19 03:28:23	成功	帮助
homebrew	系统	2018-09-19 01:34:36	成功	帮助
scientificlinux	系统	2018-09-18 22:07:37	同步中	帮助
sabayonlinux	系统	2018-09-19 03:03:03	成功	
archlinux	系统	2018-09-19 02:02:41	成功	帮助
deepin	系统	2018-09-19 03:05:12	成功	帮助
opensuse	系统	2018-09-19 04:33:08	失败	帮助

找到 `Ubuntu` 点击帮助, 复制 `Ubuntu 18` 的配置源, 并粘贴到 WSL bash 窗口中



ubuntu 18.04(bionic) 配置源如下

```
**[terminal]
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
```

按 `Esc` 键退出插入模式, 输入 `:wq` 保存并退出

依次输入下面两条命令更新软件

```
**[terminal]
**[command sudo apt update]
**[command sudo apt upgrade -y]
```

```
musk@musk-OptiPlex-5090:~$ sudo apt update
[sudo] password for musk:
Get:1 http://mirrors.aliyun.com/ubuntu bionic InRelease [242 kB]
Get:2 http://mirrors.aliyun.com/ubuntu bionic-security InRelease [83.2 kB]
Get:3 http://mirrors.aliyun.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://mirrors.aliyun.com/ubuntu bionic-proposed InRelease [242 kB]
Get:5 http://mirrors.aliyun.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:6 http://mirrors.aliyun.com/ubuntu bionic/universe Sources [9051 kB]
Get:7 http://mirrors.aliyun.com/ubuntu bionic/main Sources [829 kB]
Get:8 http://mirrors.aliyun.com/ubuntu bionic/multiverse Sources [181 kB]
Get:9 http://mirrors.aliyun.com/ubuntu bionic/restricted Sources [5324 B]
Get:10 http://mirrors.aliyun.com/ubuntu bionic/main amd64 Packages [1019 kB]
Get:11 http://mirrors.aliyun.com/ubuntu bionic/main Translation-en [516 kB]
Get:12 http://mirrors.aliyun.com/ubuntu bionic/restricted amd64 Packages [9184 B]
Get:13 http://mirrors.aliyun.com/ubuntu bionic/restricted Translation-en [3584 B]
Get:14 http://mirrors.aliyun.com/ubuntu bionic/universe amd64 Packages [8570 kB]
Get:15 http://mirrors.aliyun.com/ubuntu bionic/universe Translation-en [4941 kB]
Get:16 http://mirrors.aliyun.com/ubuntu bionic/multiverse amd64 Packages [151 kB]
Get:17 http://mirrors.aliyun.com/ubuntu bionic/multiverse Translation-en [108 kB]
Get:18 http://mirrors.aliyun.com/ubuntu bionic-security/multiverse Sources [1336 B]
Get:19 http://mirrors.aliyun.com/ubuntu bionic-security/main Sources [47.0 kB]
Get:20 http://mirrors.aliyun.com/ubuntu bionic-security/universe Sources [17.3 kB]
Get:21 http://mirrors.aliyun.com/ubuntu bionic-security/main amd64 Packages [167 kB]
Get:22 http://mirrors.aliyun.com/ubuntu bionic-security/main Translation-en [62.9 kB]
Get:23 http://mirrors.aliyun.com/ubuntu bionic-security/universe amd64 Packages [66.6 kB]
Get:24 http://mirrors.aliyun.com/ubuntu bionic-security/universe Translation-en [39.2 kB]
Get:25 http://mirrors.aliyun.com/ubuntu bionic-security/multiverse amd64 Packages [1444 B]
Get:26 http://mirrors.aliyun.com/ubuntu bionic-security/multiverse Translation-en [996 B]
Get:27 http://mirrors.aliyun.com/ubuntu bionic-updates/multiverse Sources [3216 B]
Get:28 http://mirrors.aliyun.com/ubuntu bionic-updates/main Sources [166 kB]
```

2. 缩短显示路径

当进入某个文件路径的时候,由于Linux终端命令行默认显示文件所在的全部路径,有时过长,看着很不舒服,所以我们可以配置,让其仅仅显示当前文件所在目录名称.

配置步骤：

1.修改 `.bashrc` 文件(用户根目录下) `vim` 打开 `.bashrc` 文件,

```
**[terminal]
**[command vim ~/.bashrc]
```

找到如下这行 (以实际代码所在行为准) :

```
**[terminal]
PS1='${debian_chroot:+($debian_chroot)}[\u033[01;32m]\u@\\h[\u033[00m]:[\u033[01;34m]\\w[\u033[00m]\$ '
```

```
50   11
57 fi
58
59 if [ "$color_prompt" = yes ]; then
60   PS1='${debian_chroot:+($debian_chroot)}[\u033[01;32m]\u@\\h[\u033[00m]:[\u033[01;34m]\\w[\u033[00m]\$ '
61 else
62   PS1='${debian_chroot:+($debian_chroot)}\u@\\h:\\w\$ '
63 fi
64 unset color_prompt force_color_prompt
```

将上面这行代码中的小写 `w` 改为大写的 `W`, 保存退出 (`wq`)

2.在终端下执行 `source ~/.bashrc` 使其生效

完成上面两步操作之后,重新进入到一个目录,此时在终端下只显示当前文件所在目录名称.

更改前效果 :

```
**[terminal]
**[prompt musk@ASUS-Laptop]**[path /etc/vim]**[delimiter $]
```

更改后效果 :

```
**[terminal]
**[prompt musk@ASUS-Laptop]**[path vim]**[delimiter $]
```

3.VIM 编辑器默认显示行号

1.打开 WSL Bash ,使用 `vim` 编辑配置文件

```
**[terminal]
**[command sudo vim /etc/vim/vimrc]
```

2.在文件中任意位置添加一行

```
**[terminal]
**[command set number]
```

也可缩写成

```
**[terminal]
**[command set nu]
```

4. 安装 screefetch 显示系统信息(装X工具)

```
**[terminal]
```

```
**[command sudo apt install screenfetch -y]
```

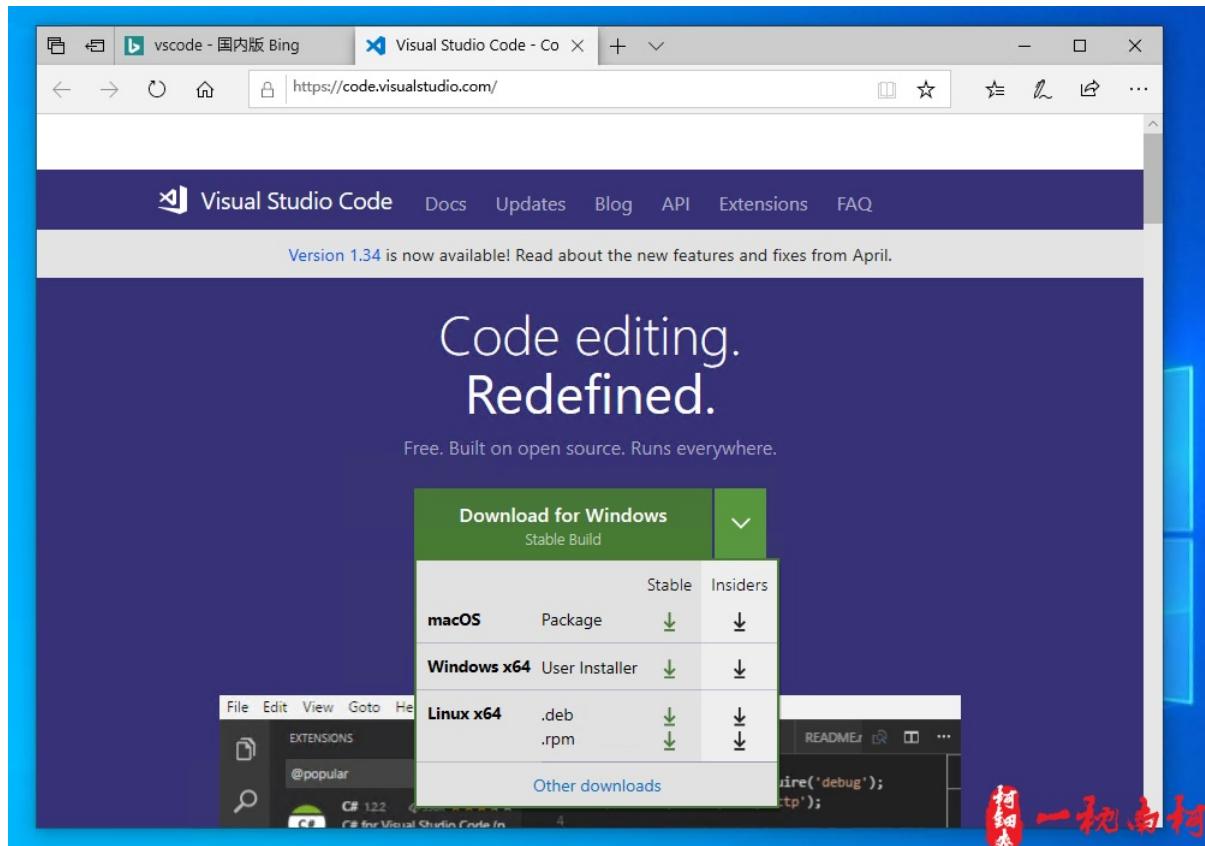
四、在 Visual Studio Code 中使用 WSL 做为默认终端

在推荐 WSL 的同时,我要向你推荐一款在微软拥抱开源世界之后推出的 开源 、 强大 、 轻量 的全能编辑器 —— `vs Code` , 来搭配 WSL 作为我们日常开发环境;

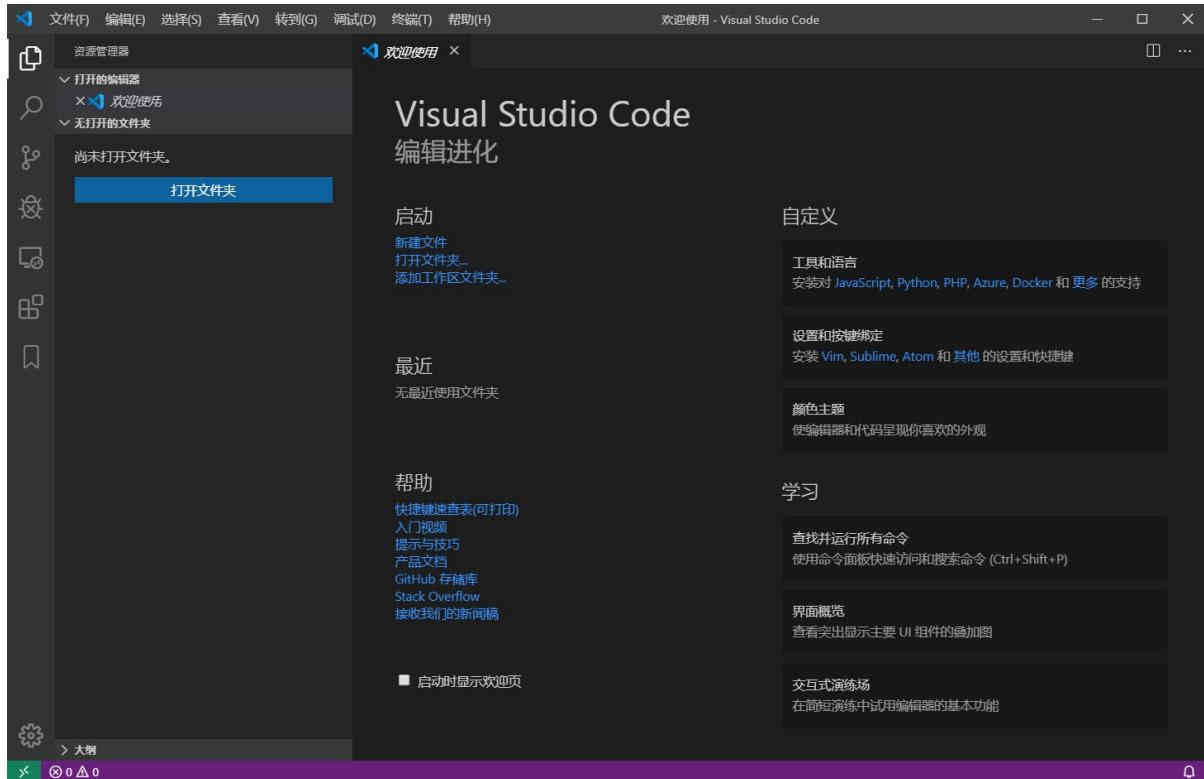
那么我们接下来就安装 `vsCode` 并将 WSL 配置为 VSCode 的默认终端

- 下载安装 `vsCode`

网页搜索 `vscode` , 打开 [VSCode 官网](https://code.visualstudio.com/), 我们这里选择下载 windows x64

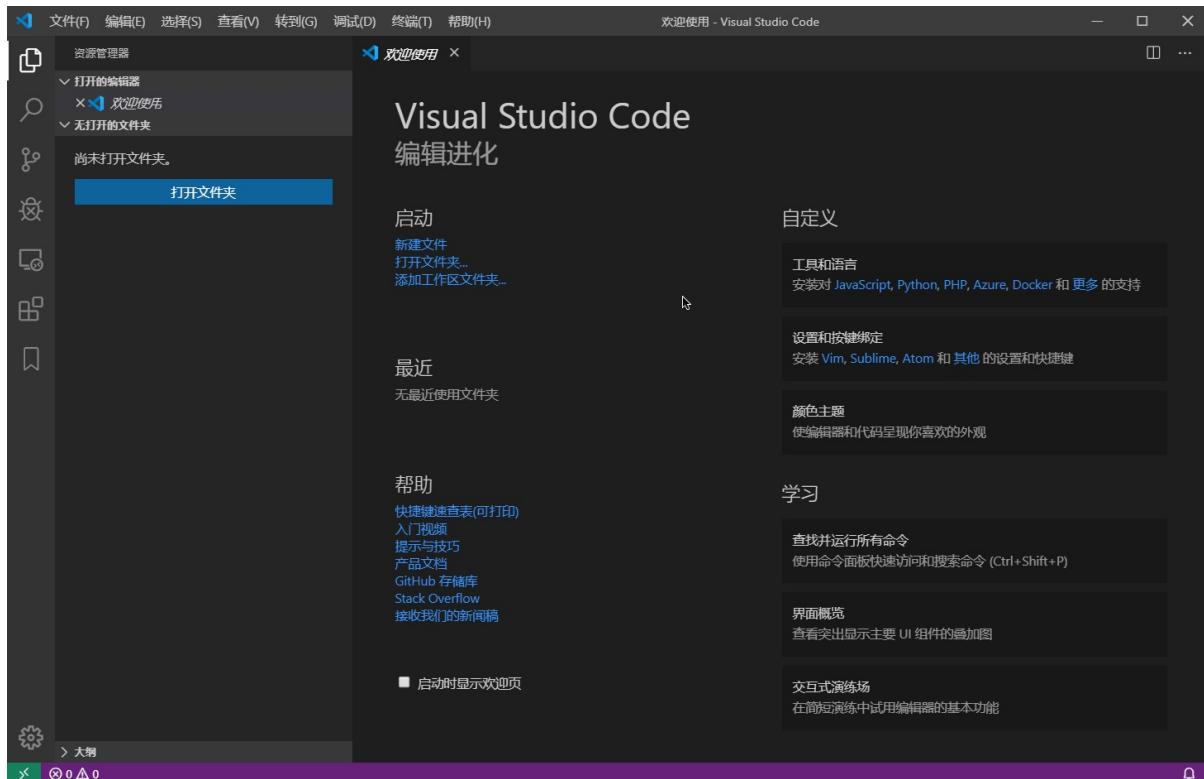


根据安装提示的完成安装即可(具体安装细节请自行 Google)



- 将 VS Code 终端配置为 wsl

在 Windows 上 vs code 默认终端采用的 shell 为 CMD , 我们这里将其切换为 wsl



Ctrl+Shift+P 打开命令执行栏,输入 shell 选择 wsl , `Ctrl+`` 即可打开 VS Code 终端窗口.

- 更多 vs code 使用技巧请自行 Google

- 一、前言
- 二、下载并配置 ESP32 编译工具链及SDK
 - 1. 下载 ESP32 SDK : ESP-IDF
 - 2. 下载并解压 ESP32 编译工具链
 - 3. 安装依赖软件包
 - 4. 配置系统环境变量(WSL)
- 三、VScode 编译运行/调试 ESP32
 - 在 VSCode 中编译运行 HelloWorld
 - 配置 ESP32 GDB 调试工具
 - 运行调试 ESP32 (Jlink/GDB)

一、前言

摘要:

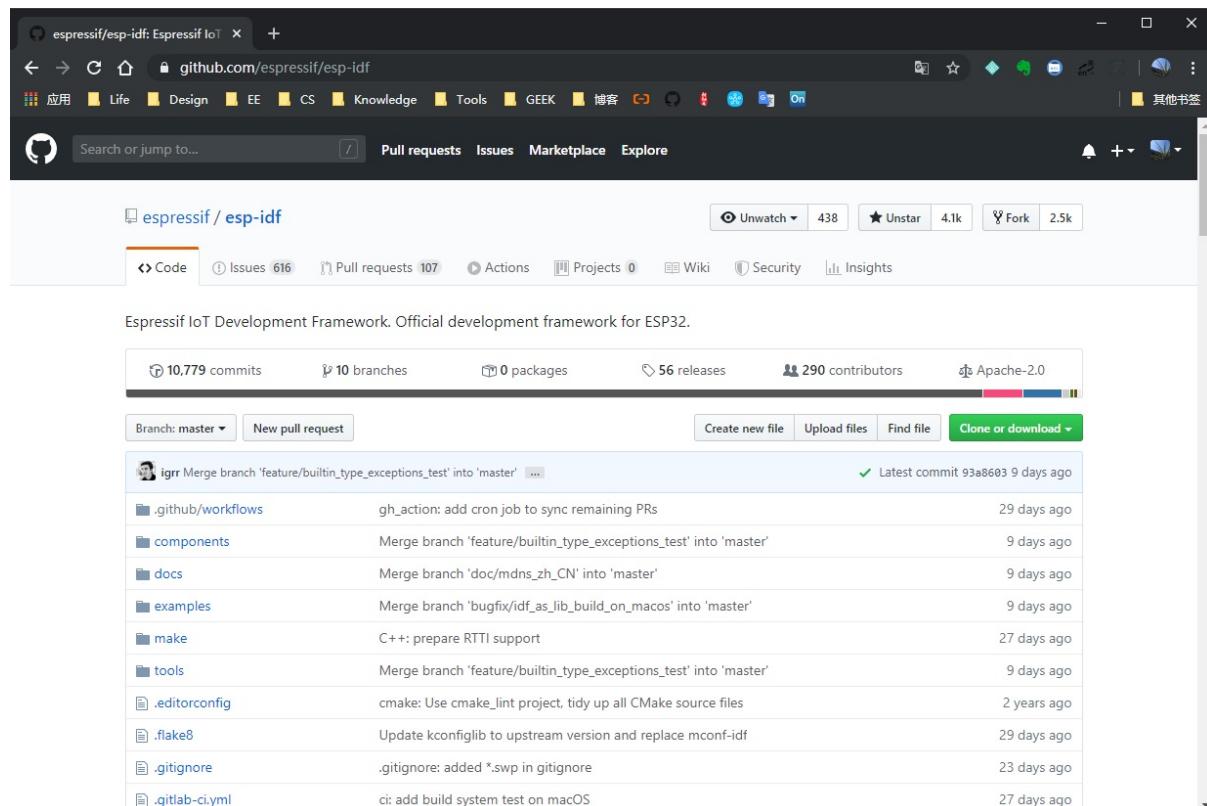
在前面我们已经安装好了 WSL 以及 VS Code, 那么下面我们就搭建 ESP32 的开发环境

二、下载并配置 ESP32 编译工具链及SDK

1. 下载 ESP32 SDK : ESP-IDF

- 从 Github 下载

打开乐鑫官方 ESP-IDF 的 GitHub 页面: [Espressif/esp-idf: https://github.com/espressif/esp-idf](https://github.com/espressif/esp-idf)



复制 Repositories 地址, 打开 Terminal Clone esp-idf

添加参数 `--recursive` 确保拉取所有子模块

```
**[terminal]
git clone --recursive https://github.com/espressif/esp-idf.git
```

The screenshot shows the GitHub repository page for 'esp-idf'. At the top, there's a terminal window showing the command to clone the repository. Below it is the GitHub repository interface. A red arrow points from the bottom of the previous screenshot to the 'Clone with HTTPS' button in the top right corner of the repository page. This button is part of a dropdown menu that includes 'Clone with HTTPS' and 'Use SSH'. The URL 'https://github.com/espressif/esp-idf.git' is displayed next to the button, also enclosed in a red box.

```
git x + ~
→ f mkdir Espressif
→ f cd Espressif
→ Espressif mkdir SDK
→ Espressif cd SDK
→ SDK git clone --recursive https://github.com/espressif/esp-idf.git
Cloning into 'esp-idf'...
remote: Enumerating objects: 122734, done.
Receiving objects: 6% (7415/122734), 1.83 MiB | 7.00 KiB/s
```

我们这里使用 `git checkout` 检出 `release/V4.0` 分支进行开发,并拉取更新所有子模块

```
**[terminal]
cd esp-idf
git checkout release/V4.0
git submodule update --init --recursive
```

- 网盘下载

2. 下载并解压 ESP32 编译工具链

我们这里使用的是 [ESP-IDF release/v4.0](#) 对应的工具链版本为 2019r2, 关于编译链官方说明请查看 -> [Linux 平台工具链的标准设置\(传统 GNU Make\)](#)

Linux 版的 ESP32 工具链可以从 Espressif 的网站下载(我们这里使用 x64 版本):

64 位 Linux: https://dl.espressif.com/dl/xtensa-esp32-elf-gcc8_2_0-esp-2019r2-linux-amd64.tar.gz

32 位 Linux: https://dl.espressif.com/dl/xtensa-esp32-elf-gcc8_2_0-esp-2019r2-linux-i686.tar.gz

下载压缩文件之后, 解压到指定目录中(我这里解压到 `/mnt/f/Espressif/Toolchain/`):

64 位 Linux:

```
mkdir -p /mnt/f/Espressif/Toolchain/
cd /mnt/f/Espressif/Toolchain/
tar -xzf ./xtensa-esp32-elf-gcc8_2_0-esp-2019r2-linux-amd64.tar.gz
```

32 位 Linux:

```
mkdir -p /mnt/f/Espressif/Toolchain/
cd /mnt/f/Espressif/Toolchain/
tar -xzf ./xtensa-esp32-elf-gcc8_2_0-esp-2019r2-linux-i686.tar.gz
```

3. 安装依赖软件包

```
sudo apt-get install git wget libncurses-dev flex bison gperf python python-click python-pip python-setuptools python-serial python-cryptography python-future python-pyparsing python-pyelftools cmake ninja-build ccache
```

```
cp ✘ Musk ✘ ./c/Users/Musk ✘ + ▾ - □ ×
→ Musk sudo apt-get install git wget libncurses-dev flex bison gperf python python-click python-pip python-setuptools python-serial python-cryptography python-future python-pyparsing python-pyelftools cmake ninja-build ccache
[sudo] password for musk:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libncurses5-dev' instead of 'libncurses-dev'
bison is already the newest version (2:3.0.4.dfsg-1build1).
ccache is already the newest version (3.4.1-1).
flex is already the newest version (2.6.4-6).
python is already the newest version (2.7.15~rc1-1).
python-click is already the newest version (6.7-3).
python-future is already the newest version (0.15.2-4ubuntu2).
python-pyparsing is already the newest version (2.2.0+dfsg1-2).
python-serial is already the newest version (3.4-2).
python-setuptools is already the newest version (39.0.1-2).
gperf is already the newest version (3.1-1).
ninja-build is already the newest version (1.8.2-1).
python-pyelftools is already the newest version (0.24-4).
git is already the newest version (1:2.17.1-1ubuntu0.4).
python-cryptography is already the newest version (2.1.4-1ubuntu1.3).
wget is already the newest version (1.19.4-1ubuntu2.2).
cmake is already the newest version (3.10.2-1ubuntu2.18.04.1).
libncurses5-dev is already the newest version (6.1-1ubuntu1.18.04).
python-pip is already the newest version (9.0.1-2.3~ubuntu1.18.04.1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
→ Musk
```

3. 配置系统环境变量(WSL)

将编译链路径/ESP-IDF路径导出到系统环境变量中

```
vim ~/.profile
```

按 `i` 进入插入模式,导出工具链及 ESP-IDF 目录到系统环境变量中(工具链路径/esp-idf 路径根据实际情况填写)

- 永久生效配置环境变量

```
export PATH="/mnt/f/Espressif/Toolchain/xtensa-esp32-elf/bin:$PATH"
export IDF_PATH="/mnt/f/Espressif/SDK/esp-idf"
```

- 临时配置环境变量

```
alias ESP32='export PATH="/mnt/f/Espressif/Toolchain/xtensa-esp32-elf/bin:$PATH"'
alias ESP32_IDF='export IDF_PATH="/mnt/f/Espressif/SDK/esp-idf"'
```

按 `esc` 退出到命令行模式,输入 `:wq` 回车(w:wirte q: quit) 写入文件并退出

立即生效更改(否则需要重新打开 Terminal)

```
**[terminal]
*[command source ~/.profile]
```

永久生效/临时配置的区别:

永久生效配置不必在每次重新打开 Terminal 时,输入 `ESP32 / ESP32_IDF` 来生效 ESP32 开发环境 ,但相应的缺点就是每次当你在两个版本的 SDK 的切换开发时需要去修改 `~/.profile` 文件; 如果在 `~/.profile` 如下配置即可使用 `ESP32_V3 / ESP32_V4` 去切换编译链版本; 使用 `ESP32_IDF_V3 / ESP32_IDF_V4` 去切换 SDK 版本.

```
**[terminal]
alias ESP32_V3='export PATH="/mnt/f/Espressif/Toolchain/xtensa-esp32-elf_v3/bin:$PATH"'
alias ESP32_IDF_V3='export IDF_PATH="/mnt/f/Espressif/SDK/esp-idf_v3"'

alias ESP32_V4='export PATH="/mnt/f/Espressif/Toolchain/xtensa-esp32-elf_v4/bin:$PATH"'
alias ESP32_IDF_V4='export IDF_PATH="/mnt/f/Espressif/SDK/esp-idf_v4"'
```

三、VScode 编译运行/调试 ESP32

在 VSCode 中编译运行 HelloWorld

打开 VSCode, `Ctrl+Shift+P` 输入 `shell`, 选择 `WSL`

编译工程:

```
make
```

完整编译工程:

```
make all
```

指定线程数编译工程:

```
make -j[N]
```

最大线程数编译工程:

```
make -j
```

编译并下载到指定串口:

```
make -j flash ESPPORT=/dev/ttySX
```

编译下载到指定串口并查看串口输出:

```
make -j flash monitor ESPPORT=/dev/ttySX
```

配置 ESP32 GDB 调试工具

运行调试 ESP32 (Jlink/GDB)

Copyright © qinun575@foxmail.com 2019 all right reserved , powered by Gitbook 该文件修订时间 : 2019-11-29 08:57:00

- 一、前言
- 二、下载安装 ESP-IDF-Tools-Setup
 - 使用 ESP-IDF 安装器安装 ESP32 依赖环境
- 三、VScode 编译运行 ESP32 例程
 - 在 VSCode 中编译运行 HelloWorld

一、前言

摘要:

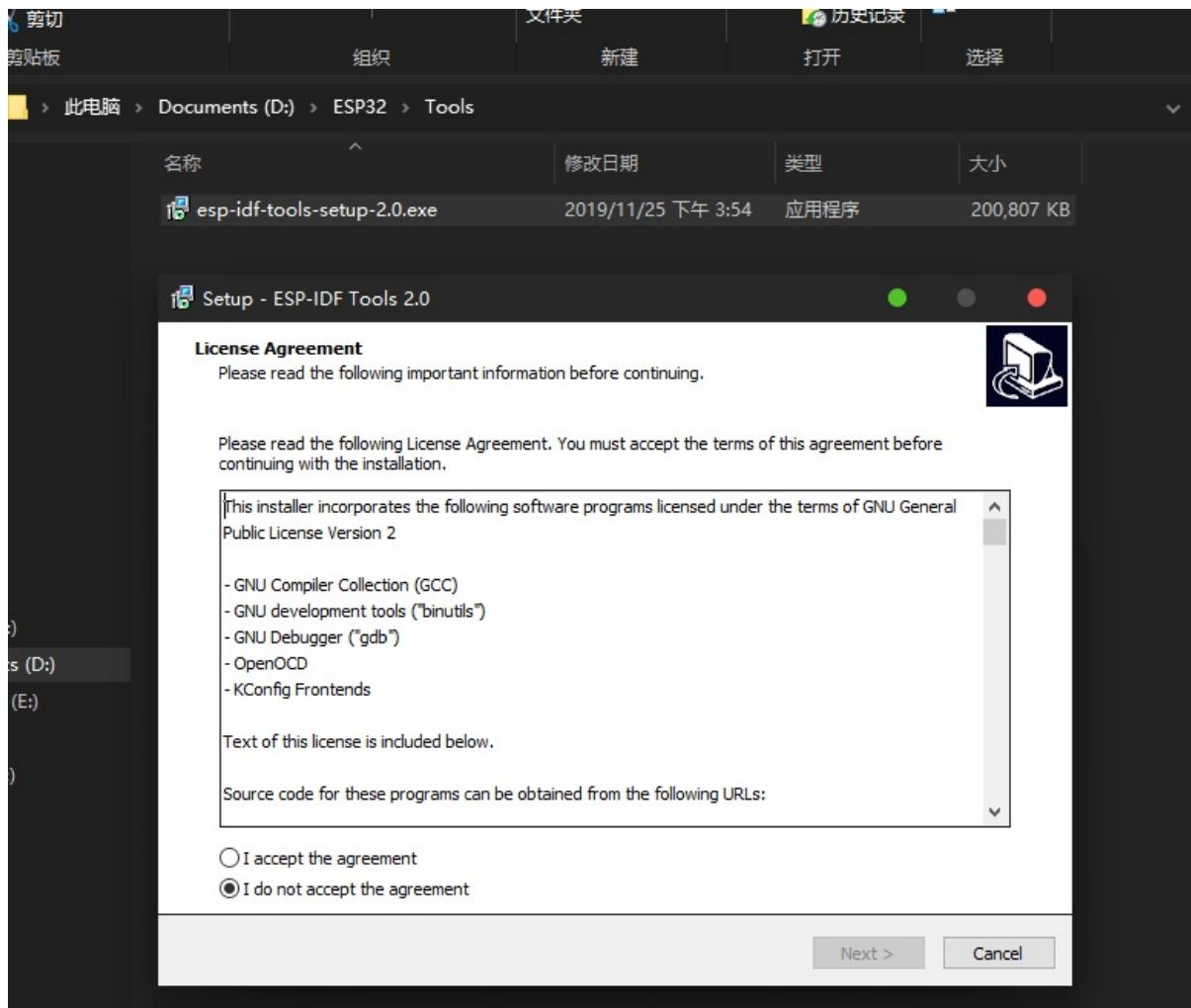
由于大部分嵌入式/MCU 开发工程师都是在 Windows 进行开发的,所以乐鑫推出了适用于 Windows 上搭建 ESP32 开发环境的安装器 ESP-IDF 工具安装器(ESP-IDF-tools-setup) 我们可以使用 ESP-IDF 安装器 + VS Code 搭建 ESP32 开发环境 来快速搭建 ESP32 开发环境

二、下载安装 ESP-IDF-Tools-Setup

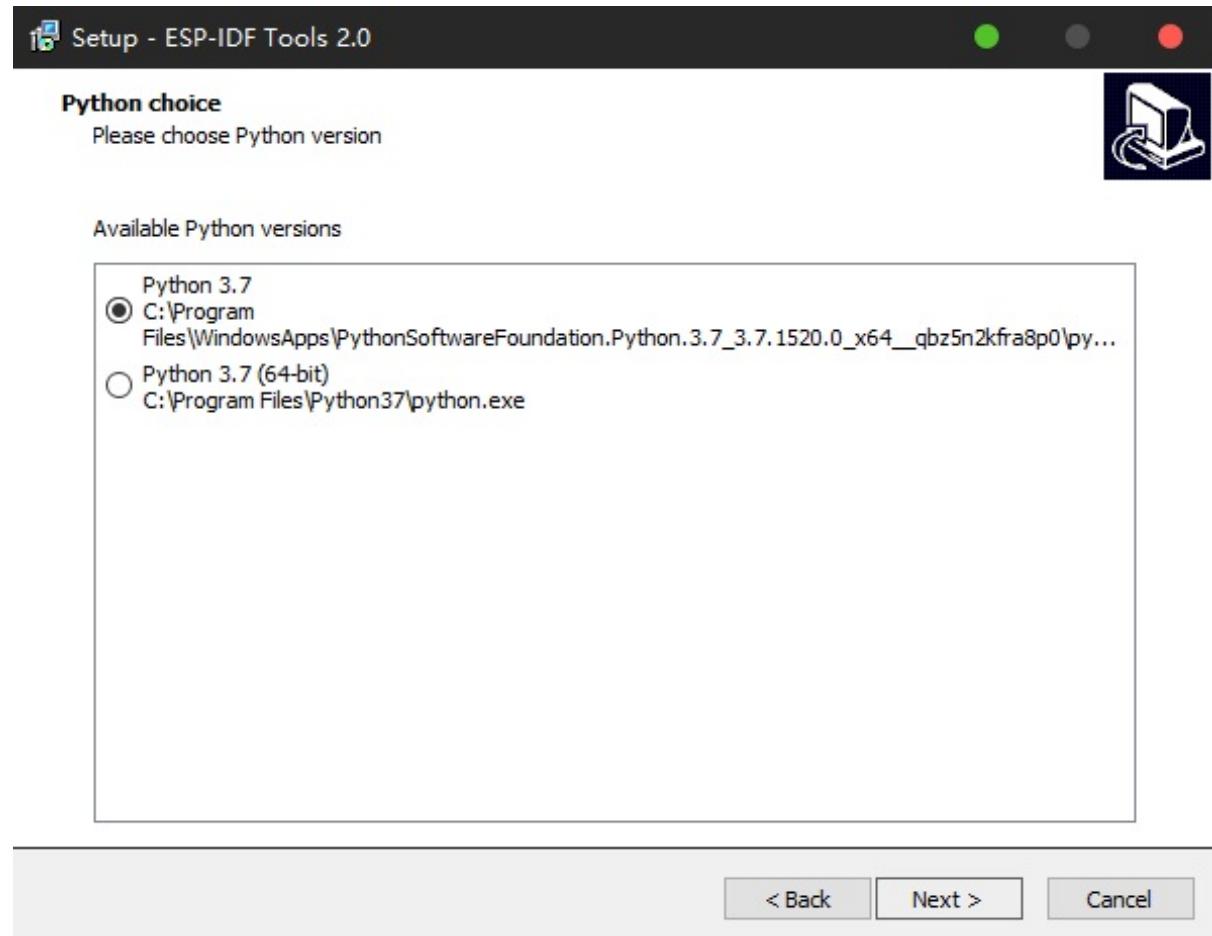
ESP-IDF 安装器 下载地址: <https://dl.espressif.com/dl/esp-idf-tools-setup-2.0.exe>

使用 ESP-IDF 安装器安装 ESP32 依赖环境

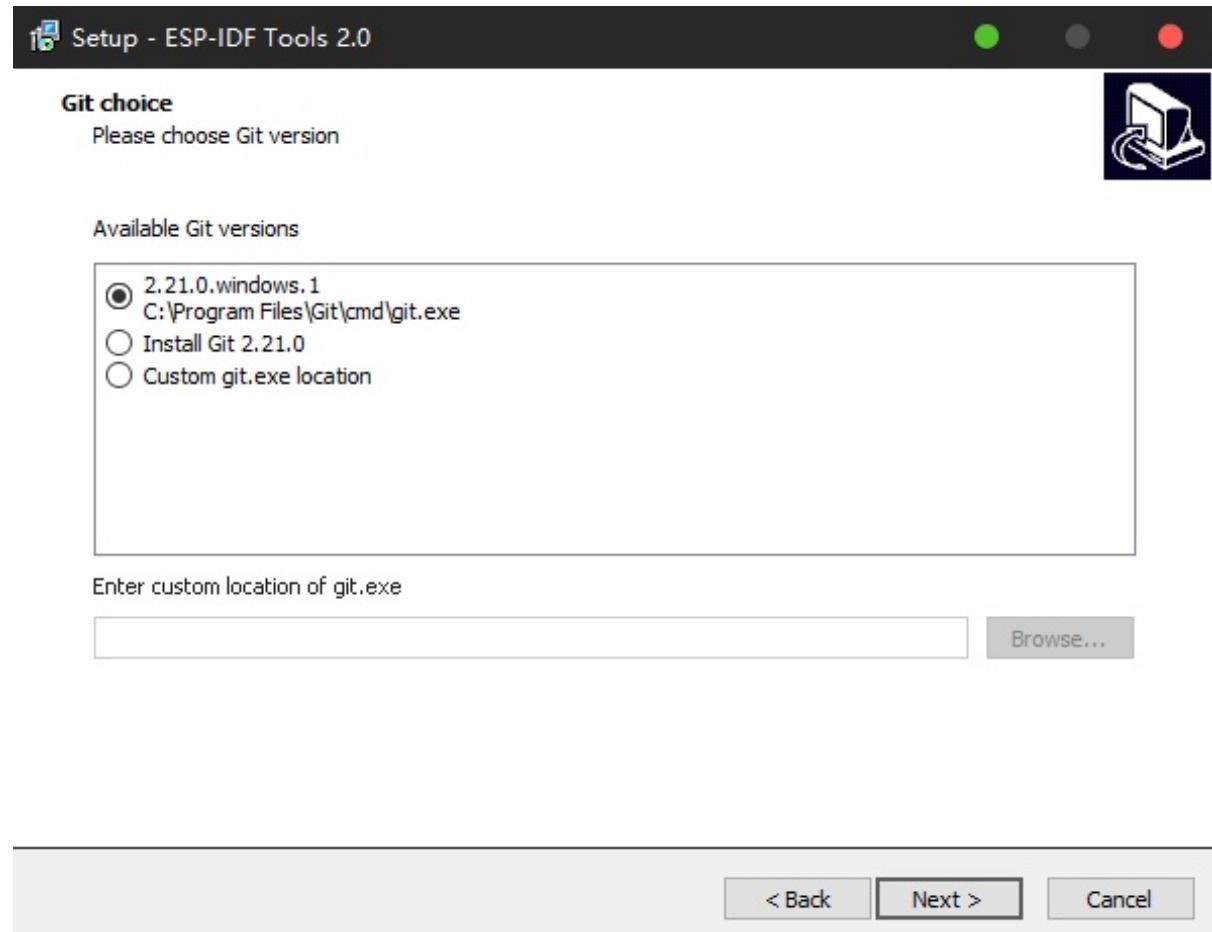
1. 以管理员身份运行 `esp-idf-tools-setup-2.0.exe`
2. 接受许可



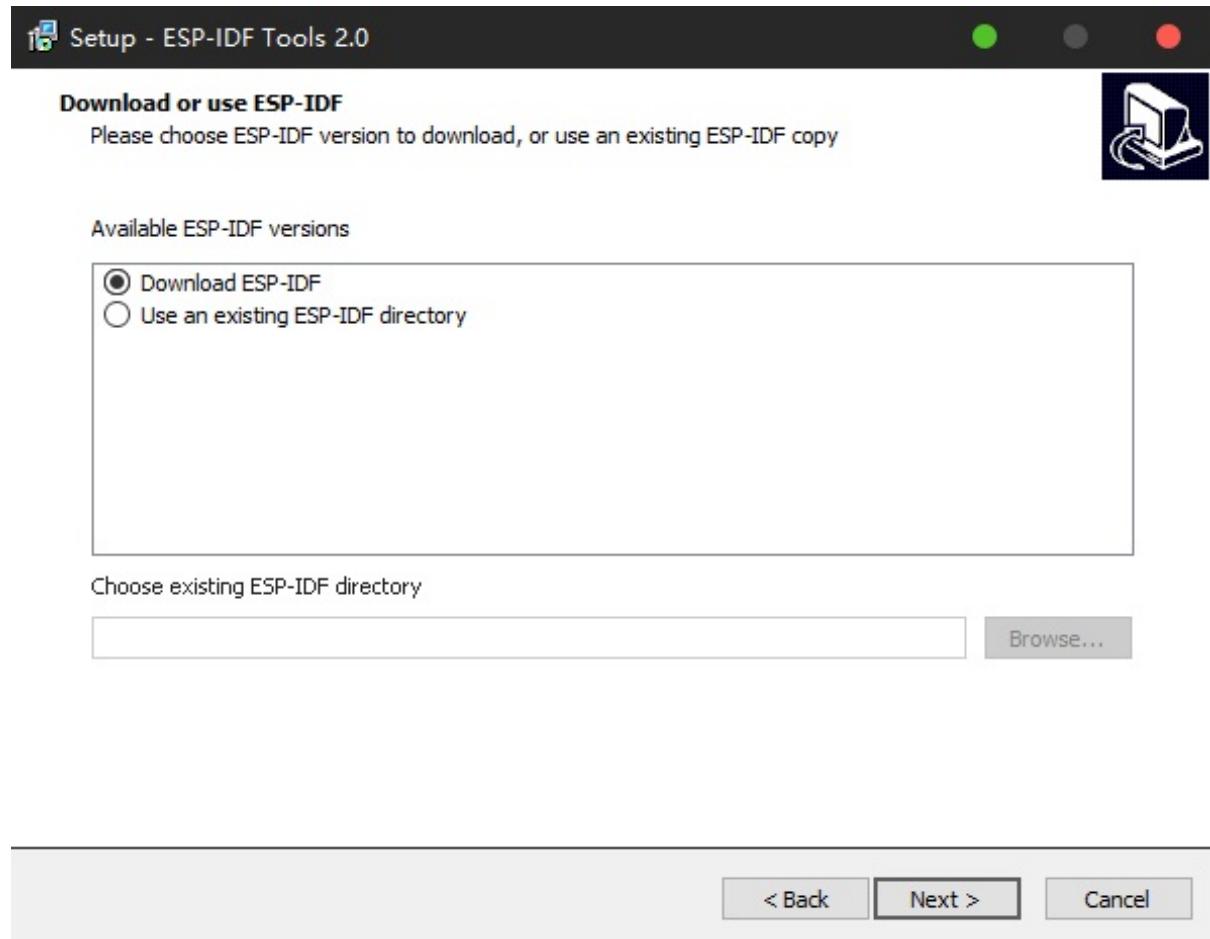
1. 选择安装 Python



1. 选择安装 git



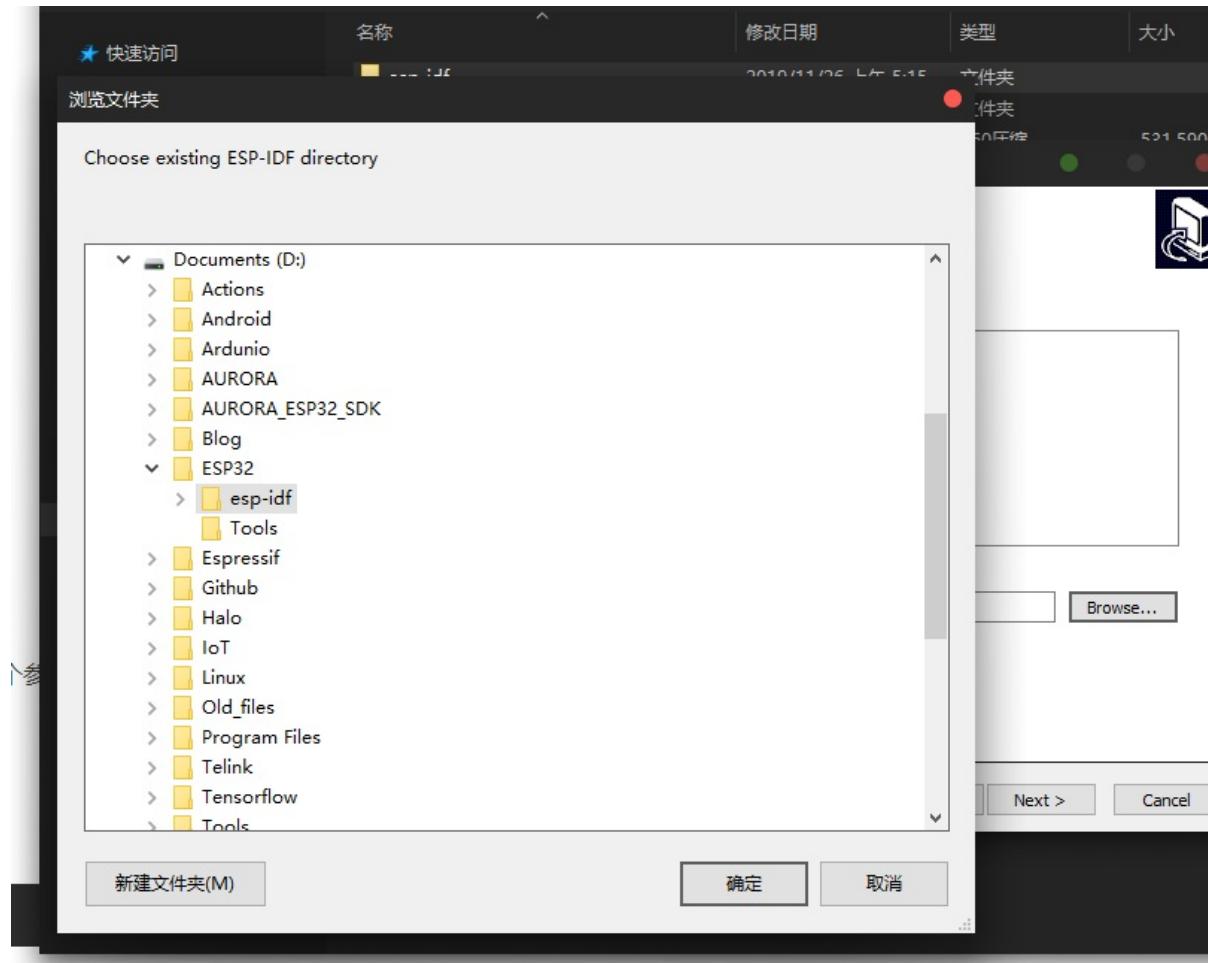
1. 下载/选择 ESP-IDF 目录



由于我们这里使用的是 Release/V4.0 版本,安装器会从 GitHub 上下载 esp-idf, 而 GitHub 由于某些原因在国内的下载速度异常感人,所以我们这里从 gitee(码云)/百度云 下载 esp-idf 的镜像

百度云下载(链接 : <https://pan.baidu.com/s/1CCkClgQ1eu9vJ-o5Rx2sCQ> 提取码 : bd30)

下载之后使用 360压缩软件 解压到指定目录,然后使用 ESP-IDF 安装器选定该目录

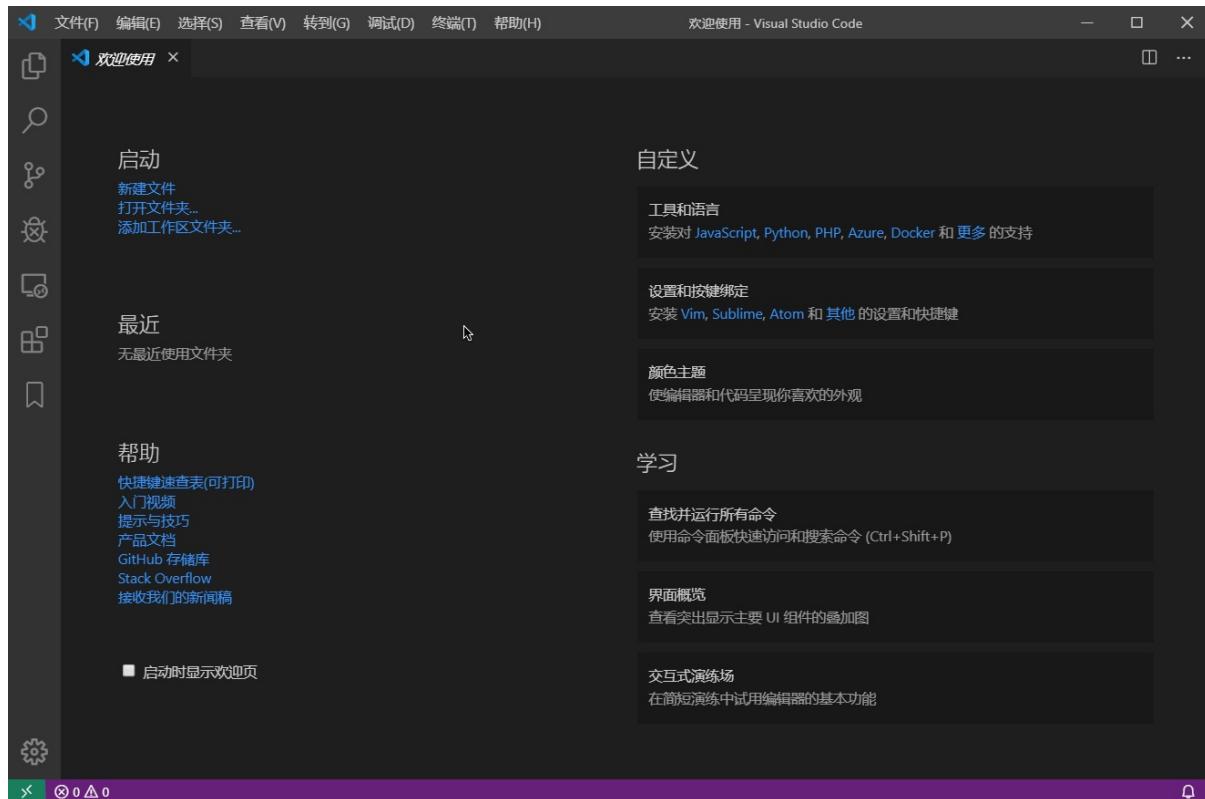


三、VScode 编译运行 ESP32 例程

1. 下载并安装 VSCode, 并将配置终端为 CMD, 及配置终端参数

Ctrl+Shift+P 打开命令面板, 输入 `shell` 选择 `CMD` 然后回车, `Ctrl+`` 即可打开 VS Code 终端窗口.

```
{
  "terminal.integrated.shell.windows": "C:\\Windows\\System32\\cmd.exe",
  "terminal.integrated.shellArgs.windows": [
    "/k",
    " D:\\ESP32\\.espressif\\idf_cmd_init.bat C:\\Program Files\\Python37 C:\\Program Files\\Git\\cmd"
  ]
}
```



编译示例工程

idf.py 相关命令

查看 idf.py 支持命令: `idf.py --help`

修改工程配置: `idf.py menuconfig`

编译工程: `idf.py build`

全编译工程(重新编译): `idf.py all`

编译下载到指定串口: `idf.py -p COM1 flash`

编译下载并查看: `idf.py -p COM1 flash monitor`

在 VSCode 中编译运行 HelloWorld

在 ESP-IDF 目录中右键选择 "

Copyright © qinun575@foxmail.com 2019 all right reserved , powered by Gitbook 该文件修订时间 : 2019-11-29 08:57:00

- 开发工具 - 常用 shell 命令/脚本
 - 基本 shell 操作
 - 软件/命令查看手册 man (Manual, 男人的手册)
 - 文件/路径管理
 - Ubuntu 软件包管理 - apt
 - 常用软件包
 - 解/压缩文件管理 - tar && zip
 - 文本编辑器 vi,vim,nano

开发工具 - 常用 shell 命令/脚本

本文记录在开发过程中常用的 shell 命令及脚本

基本 shell 操作

软件/命令查看手册 man (Manual, 男人的手册)

- man

文件/路径管理

- 创建文件/文件夹 touch,mkdir,echo
- 查看文件 cat,head,tail
- 文件/文件夹移动 && 复制 mv, cp
- 文件/文件夹删除 rm

Ubuntu 软件包管理 - apt

- 更新软件仓库

```
**[terminal]
**[command sudo apt update]
```

- 更新软件

```
**[terminal]
**[command sudo apt upgrade]
```

- 移除无效软件包

```
**[terminal]
**[command sudo apt autoremove]
```

常用软件包

解/压缩文件管理 - tar && zip

文本编辑器 vi, vim, nano

Copyright © qinun575@foxmail.com 2019 all right reserved , powered by Gitbook 该文件修订时间 : 2019-11-29 08:57:00