

在Ubuntu中搭建嵌入式Linux开发环境

百问网已经制作好了完备的Ubuntu镜像，可以从这里下载：

链接: https://pan.baidu.com/s/1vw4VUV_Mvt0HXz8IC66ACg
提取码: iftb

如果网盘链接无效了，可以加QQ群联系我们：341014981

我们也正在(2022.10.17开始)使用纯粹的Ubuntu环境开始教驱动入门，免费的，感兴趣者也加上面的群。

如果你使用我们的Ubuntu镜像，那么可以略过前面第1~6章的内容，直接从第7章开始看。

1. 安装基本开发工具

```
git clone https://e.coding.net/weidongshan/DevelopmentEnvConf.git
cd DevelopmentEnvConf
sudo ./Configuring_ubuntu.sh
```

2. 安装vscode

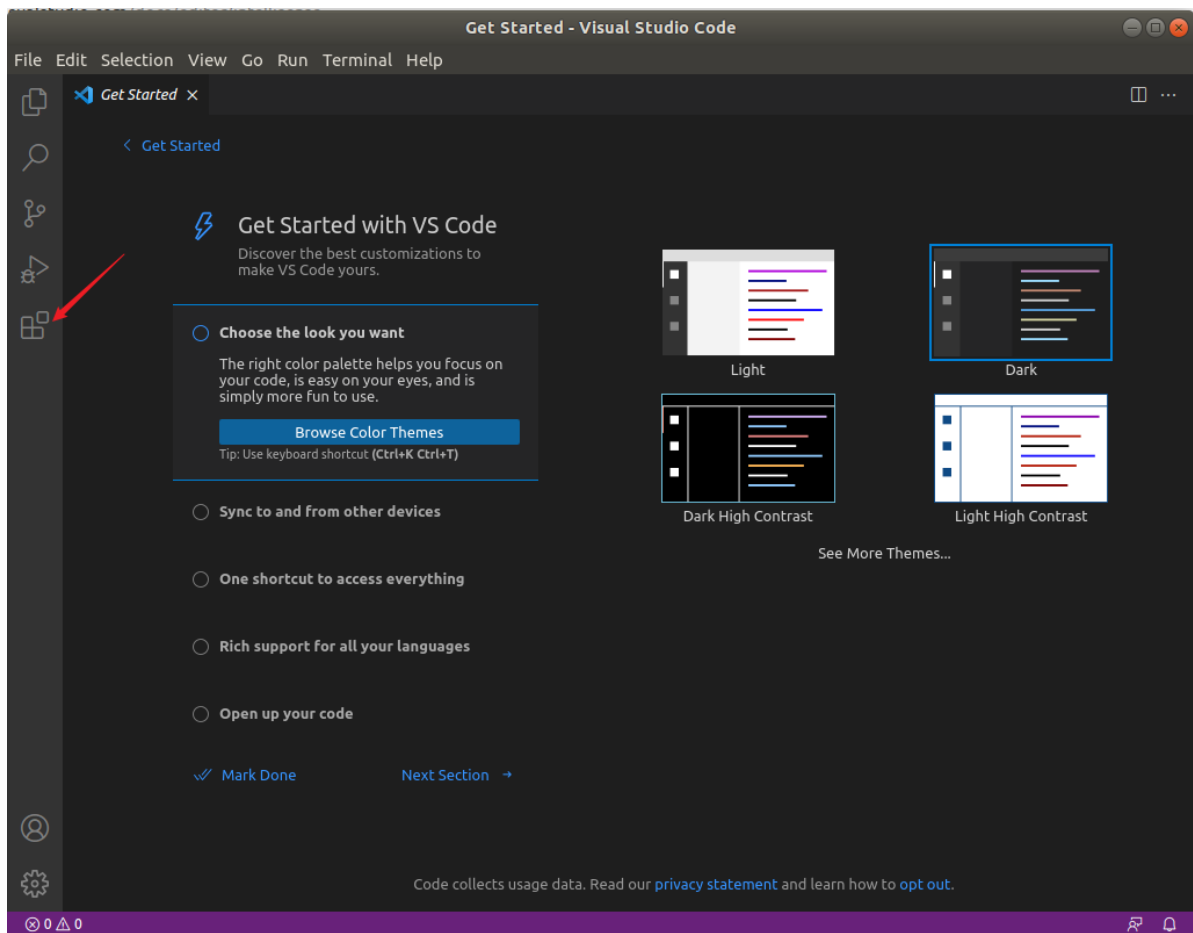
2.1 从官网下载安装

在Ubuntu中使用浏览器从从<https://code.visualstudio.com/>下载deb包，执行：

```
cd /home/book/Downloads
sudo dpkg -i code_1.72.2-1665614327_amd64.deb
```

2.2 安装插件

打开vscode后，点击左侧图标：



依次输入下列插件名字，安装：

- C/C++
- C/C++ Extension Pack
- C/C++ Snippets
- Code Runner
- Code Spell Checker
 - vscode-icons
- compareit
- DeviceTree
- Tabnine AI Autocomplete
- Bracket Pair Colorization Toggler
- Rainbow Highlighter
 - 高亮文字：shift + alt + z
 - 取消高亮：shift + alt + a
- Arm Assembly
- Chinese
- Hex Editor
- One Dark Pro
- Clangd
- Markdown All in One
- Markdown Preview Enhanced

我们已经安装的插件有这些：



EXTENSIONS



Search Extensions in Marketplace

INSTALLED

16

**Arm Assembly**Arm assembly syntax support for Visual Studio Co...
dan-c-underwood**Bracket Pair Colorization Toggler**Quickly toggle 'Bracket Pair Colorization' setting ...
Dzhavat Ushev**C/C++**C/C++ IntelliSense, debugging, and code browsing.
Microsoft**C/C++ Advanced Lint**An advanced, modern, static analysis extension fo...
Joseph Benden**C/C++ Snippets**Code snippets for C/C++
Harsh**Chinese (Simplified) (简体中文) Language Pack f...**

中文(简体)

Microsoft**clangd**C/C++ completion, navigation, and insights
LLVM**Code Runner**Run C, C++, Java, JS, PHP, Python, Perl, Ruby, Go, ...
Jun Han

29ms

**Code Spell Checker**Spelling checker for source code
Street Side Software

69ms

**compareit**Compare files
in4margaret

22ms

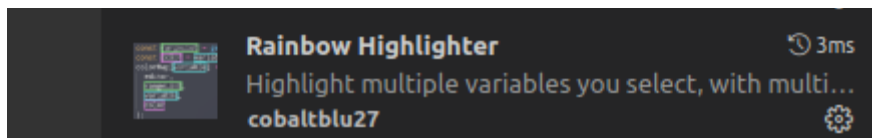
**DeviceTree**DeviceTree Language Support for Visual Studio C...
Pietro Lorefice**Hex Editor**Allows viewing and editing files in a hex editor
Microsoft**Markdown All in One**All you need to write Markdown (keyboard shortc...
Yu Zhang

22ms

**Markdown Preview Enhanced**Markdown Preview Enhanced ported to vscode
Yiyi Wang**One Dark Pro**Atom's iconic One Dark theme for Visual Studio C...
binaryify

22ms





2.3 配置clangd

2.3.1 下载clangd

前面只是安装clangd插件，它的使用还需要一个clangd程序。

我们以后使用vscode打开C文件时，会提示你安装clangd程序，它会安装最新版本(版本15)，但是这个版本有一些Bug，所以我们手工安装版本13。

在Ubuntu中使用浏览器打开<https://github.com/clangd/clangd/releases/tag/13.0.0>，下载Linux安装包：

github.com/clangd/clangd/releases/tag/13.0.0

13.0.0

github-actions released this 01 Oct 2021 13.0.0 c084539

This is the stable clangd 13.0.0 release, based on the [LLVM release sources](#).

[Release notes since clangd 12](#)

Built from [llvm/llvm-project@ d7b669b](#).

▼ Assets 8

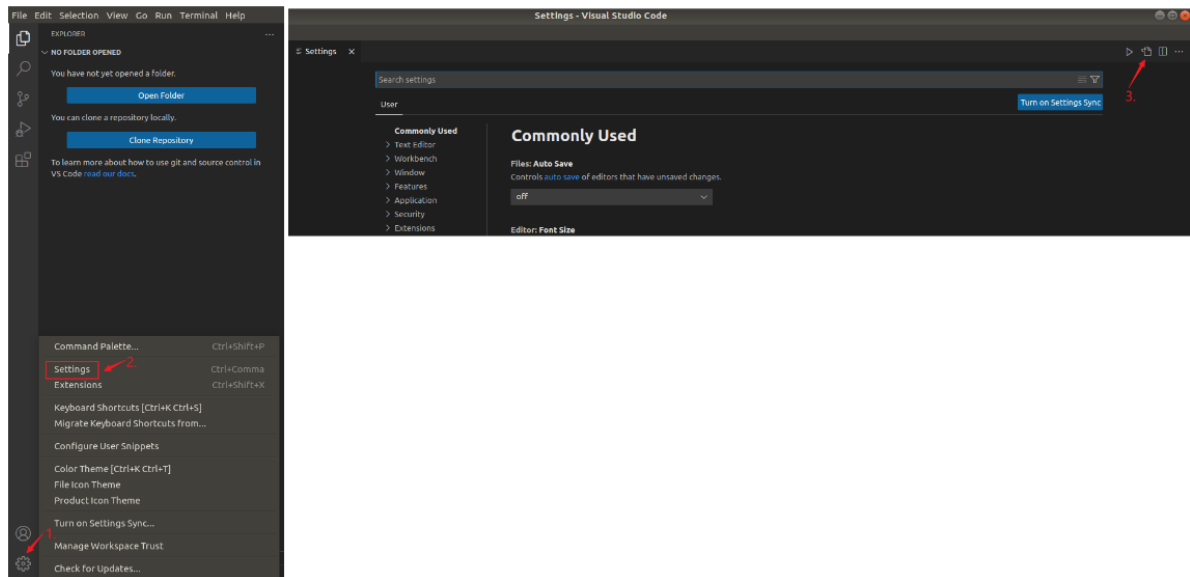
clangd-linux-13.0.0.zip	61.1 MB	01 Oct 2021
clangd-mac-13.0.0.zip	33.3 MB	01 Oct 2021
clangd-windows-13.0.0.zip	15.2 MB	01 Oct 2021
clangd_indexing_tools-linux-13.0.0.zip	85.9 MB	01 Oct 2021
clangd_indexing_tools-mac-13.0.0.zip	35.3 MB	01 Oct 2021
clangd_indexing_tools-windows-13.0.0.zip	17.5 MB	01 Oct 2021
Source code (zip)		19 Oct 2021
Source code (tar.gz)		19 Oct 2021

把下载到的clangd-linux-13.0.0.zip放到/home/book目录下，执行解压命令：

```
cd /home/book
unzip clangd-linux-13.0.0.zip
```

2.3.2 配置clangd

在vscode界面按下图步骤打开setting.json文件：



在setting.json中写入如下内容(我们第1次打开源码目录后, 这个文件可能被自动修改, 你需要再次修改它):

```
{
  "C_Cpp.default.intellisenseMode": "linux-gcc-arm",
  "C_Cpp.intellisenseEngine": "Disabled",
  "clangd.path": "/home/book/clangd_13.0.0/bin/clangd",
  "clangd.arguments": [
    "--log=verbose",
  ],
}
```

C/C++插件里的intellisense和clangd是冲突的, 如果我们没有手工设置setting.json, 当使用vscode打开C文件时也会提示禁止intellisense, 点击鼠标即可禁止。它的本质也是修改setting.json, 它会写入如下文字:

```
"C_Cpp.intellisenseEngine": "disabled",
```

上面文件有Bug, 其中的"disabled"应该改为"Disabled"。

2.3.3 安装bear

在vscode中使用clangd, 要实现精确跳转, 需要使用bear分析源码生成compile_commands.json。

执行如下命令安装:

```
sudo apt install bear
```

2.4 常用快捷键

打开C文件后，在文件里点击右键就可以看到大部分快捷键。

输入文件名打开文件：Ctrl + P
跳到某行：Ctrl + G + 行号
打开文件并跳到某行：Ctrl + p 文件名:行号
列出文件里的函数：Ctrl + Shift + O，可以输入函数名跳转
函数/变量跳转：按住Ctrl同时使用鼠标左键点击、F12
前进：~~Ctrl + Shift +~~ Alt + right
后退：~~Ctrl + Alt +~~ Alt + left
列出引用：Shift + F12
查找所有引用：~~Alt + Shift + F12~~ Shift + r
切换侧边栏展示/隐藏：Ctrl + B
打开命令菜单：Ctrl + Shift + P
手动触发建议：Ctrl + Space
手动触发参数提示：Ctrl + Shift + Space
打开/隐藏终端：Ctrl + ` (Tab上方的那个键)
重命名符号：F2
当前配置调试：F5
上/下滚编辑器：Ctrl + ↑/↓
搜索/替换：Ctrl + F/H
高亮文字：shift + alt + z
取消高亮：shift + alt + a

3. 使用WindTerm

3.1 安装WindTerm

WindTerm是Linux环境下好用的终端软件，GUI界面、支持ssh、串口等协议，可以记录历史命令。

我们使用它来打开串口操作开发板。

在Ubuntu中使用浏览器打开<https://github.com/kingToolbox/WindTerm/releases/tag/2.5.0>，下载Linux版本的软件包：

github.com/kingToolbox/WindTerm/releases/tag/2.5.0

- Auto Completion
- Auto Completion / History Commands
- Auto Completion / Quick Commands

Version 2.6.0 Prerelease 1 (2022-8-21)

If you want, you can try [WindTerm 2.6.0 Prerelease 1](#) now. Currently only the Windows x86_64 version is available.

▼ Assets 6

WindTerm_2.5.0_Linux_Portable_x86_64.tar.gz	33.7 MB	25 Jul 2022
WindTerm_2.5.0_Mac_Portable_x86_64.dmg	23.4 MB	25 Jul 2022
WindTerm_2.5.0_Windows_Portable_x86_32.zip	24.3 MB	25 Jul 2022
WindTerm_2.5.0_Windows_Portable_x86_64.zip	27.9 MB	25 Jul 2022
Source code (zip)		25 Jul 2022
Source code (tar.gz)		25 Jul 2022

把下载到的WindTerm_2.5.0_Linux_Portable_x86_64.tar.gz放到/home/book目录下，执行解压命令：

```
cd /home/book
tar xzf WindTerm_2.5.0_Linux_Portable_x86_64.tar.gz
cd WindTerm_2.5.0/
chmod +x WindTerm
```

以后就可以在桌面系统打开终端后执行以下命令打开WindTerm:

```
cd /home/book/WindTerm_2.5.0/
sudo ./WindTerm
```

3.3 配置WindTerm

WindTerm的使用方法可以参考: <https://zhuanlan.zhihu.com/p/468501270>

本节目标:

- 解决问题: 我们使用WindTerm打开/dev/ttyUSB0或/dev/ttyACM0等串口时, 不加sudo命令就会碰到权限问题
- 方便使用: 我们想在Ubuntu左侧启动栏点击鼠标就启动WindTerm

3.3.1 解决权限问题

执行如下命令把book用户放入组(dialout、tty)即可:

```
sudo newgrp dialout tty
sudo usermod -a -G dialout book
sudo usermod -a -G tty book
```

3.3.2 放入启动栏

创建一个文件 `/usr/share/applications/windterm.desktop`, 内容如下:

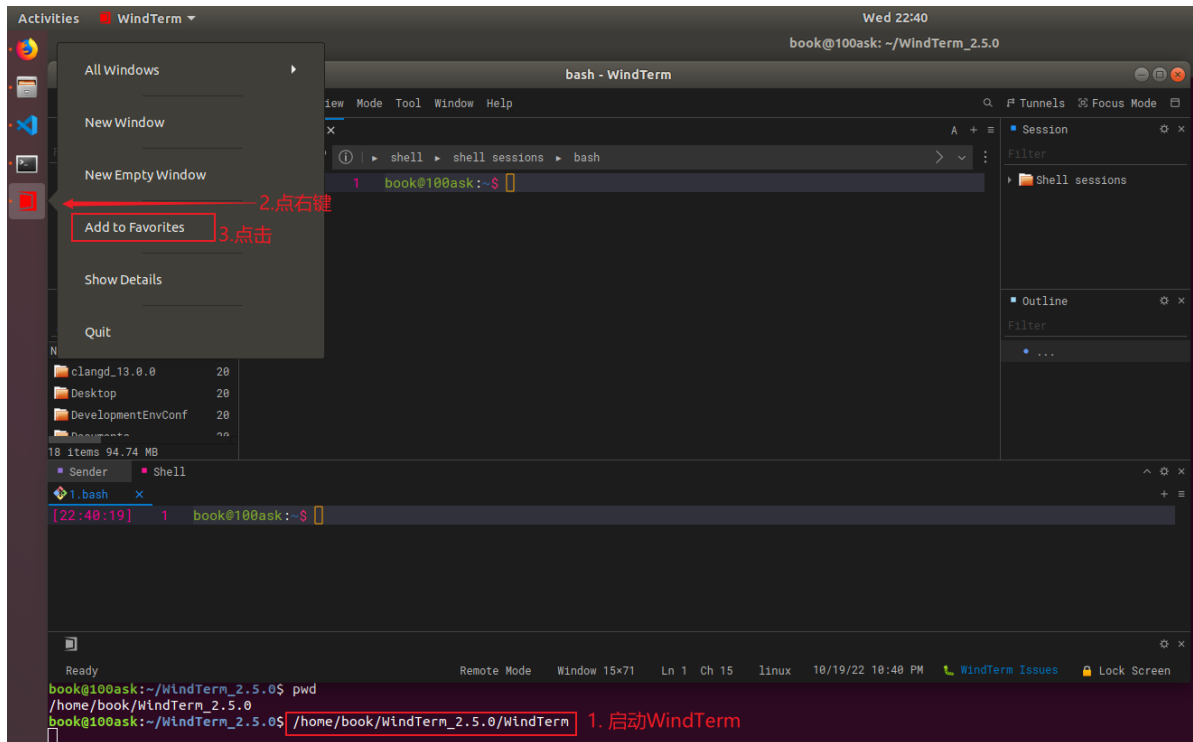
```
[Desktop Entry]
Name=WindTerm
Comment=A professional cross-platform SSH/Sftp/Shell/Telnet/Serial terminal
GenericName=Connect Client
Exec=/home/book/WindTerm_2.5.0/WindTerm
Type=Application
Icon=/home/book/WindTerm_2.5.0/windterm.png
StartupNotify=false
StartupWMClass=Code
Categories=Application;Development
Actions=new-empty-window
Keywords=windterm

[Desktop Action new-empty-window]
Name=New Empty Window
Icon=/home/book/WindTerm_2.5.0/windterm.png
Exec=/home/book/WindTerm_2.5.0/WindTerm
```


然后增加可执行权限、并启动程序：

```
sudo chmod +x /usr/share/applications/windterm.desktop  
/home/book/windTerm_2.5.0/windTerm
```

最后如下图操作，以后就可以在Ubuntu左侧的启动栏点击鼠标启动WindTerm：



4. 安装搜狗输入法

按照说明操作：<https://shurufa.sogou.com/linux/guide>

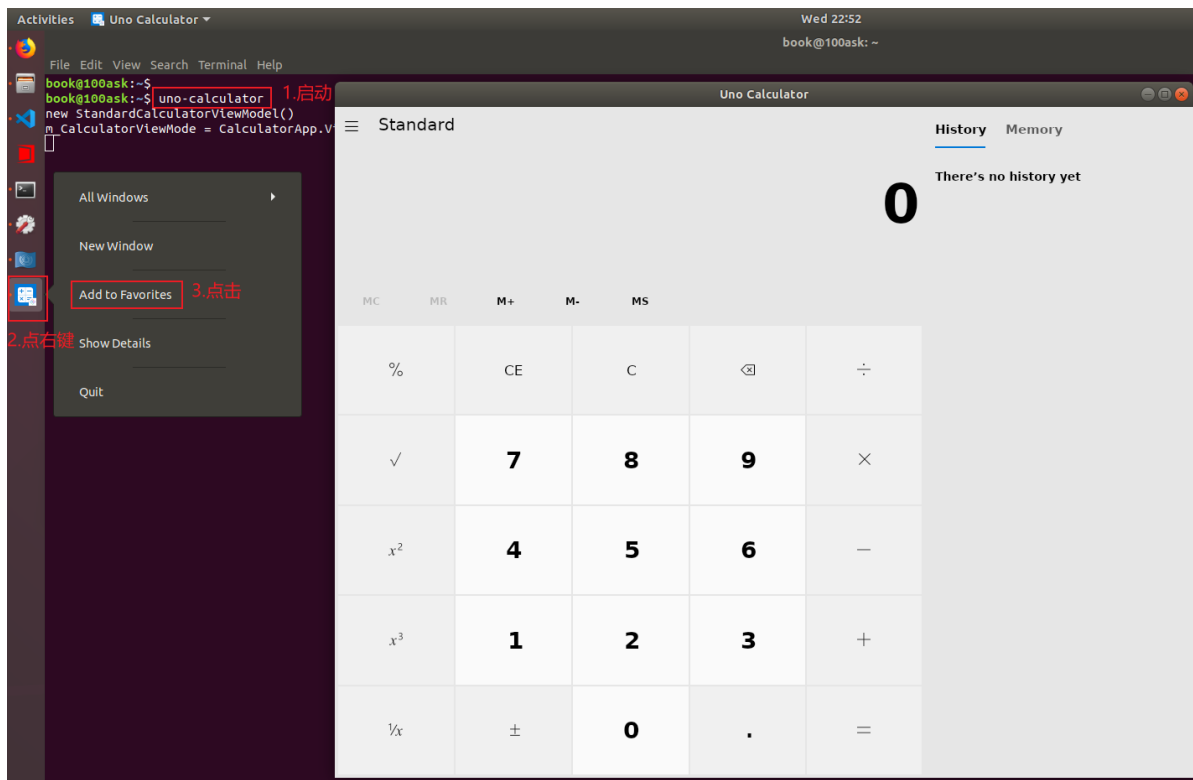
切换中英文快捷键：Ctrl+空格。

5. 安装计算器

执行如下命令安装：

```
sudo snap install uno-calculator
```

然后启动它、添加进Ubuntu左侧的启动栏，入下图操作：



6. 下载和编译内核

以百问网IMX6ULL_Pro开发板为例。

6.1 下载内核

执行如下命令：

```
$ git clone https://e.coding.net/codebug8/repo.git
$ mkdir -p 100ask_imx6ull-sdk && cd 100ask_imx6ull-sdk
$ ../repo/repo init -u https://gitee.com/weidongshan/manifests.git -b linux-sdk
-m imx6ull/100ask_imx6ull_linux4.9.88_release.xml --no-repo-verify
$ ../repo/repo sync -j4
```

6.2 配置工具链

执行如下命令：

```
gedit ~/.bashrc
```

在最后加入如下内容：

```
export ARCH=arm
export CROSS_COMPILE=arm-buildroot-linux-gnueabi-
export PATH=$PATH:/home/book/100ask_imx6ull-sdk/Toolchain/arm-buildroot-linux-
gnueabi-sdk-buildroot/bin
```

重新关闭、打开终端。

6.3 编译内核

执行如下命令：

```
$ cd /home/book/100ask_imx6ull-sdk/Linux-4.9.88
$ make 100ask_imx6ull_defconfig
$ make zImage -j4
```

7. 使用vscode阅读内核源码

7.1 生成compile_commands.json

bear命令用来生成compile_commands.json，它的用法如下：

```
bear make [其他make本身的参数]
```

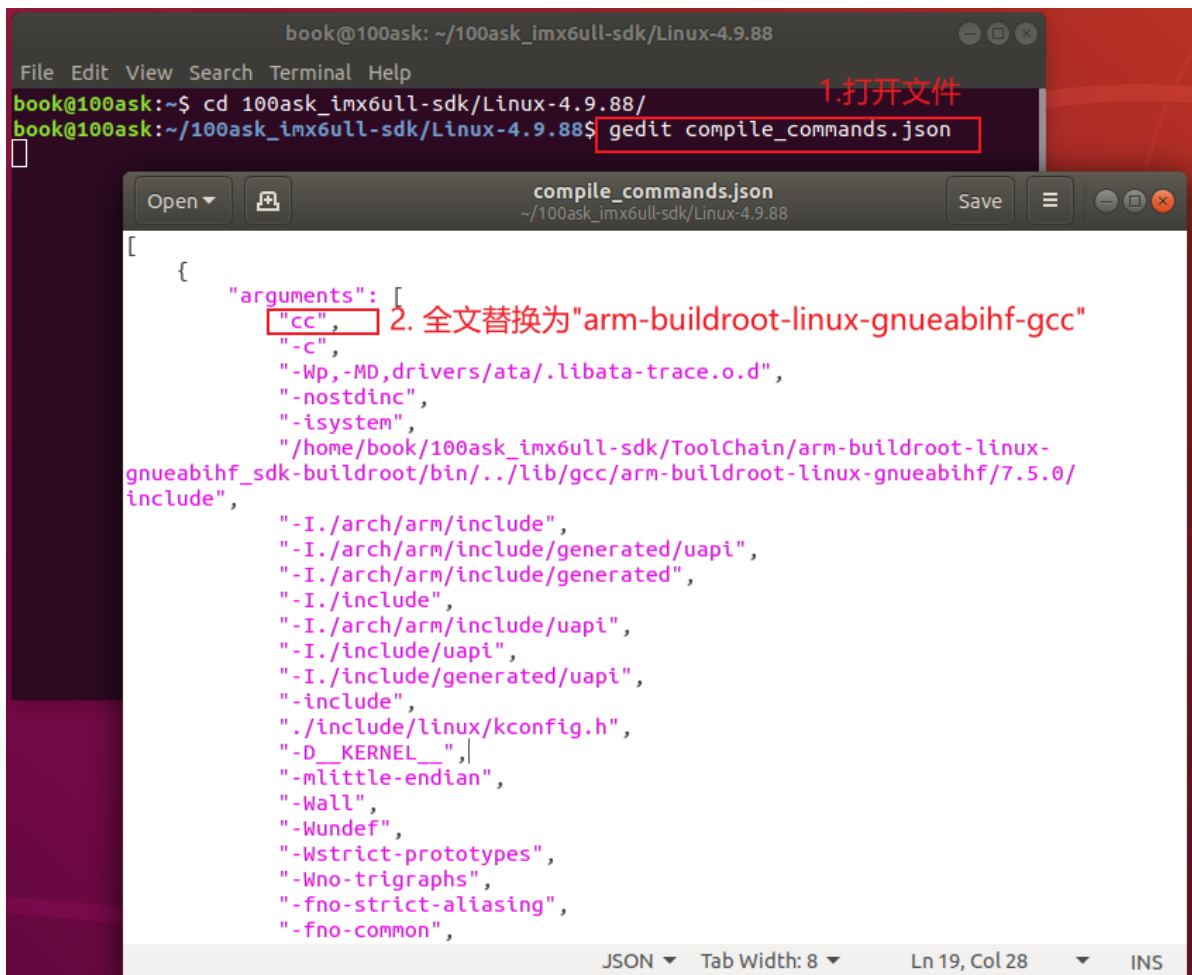
它会记录make过程编译文件时用到的命令。

编译内核时，使用如下命令：

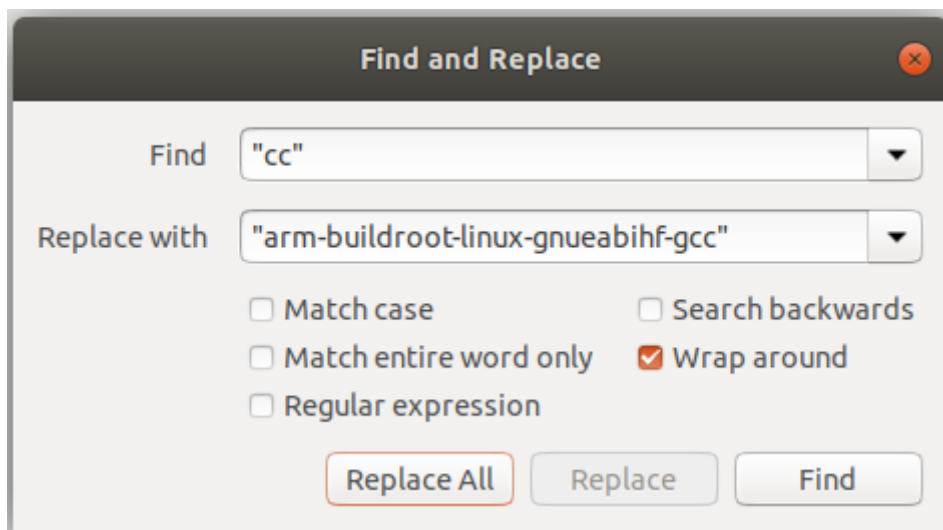
```
// 如果之前曾经编译过内核，要清除掉
make clean

// 然后重新编译
bear make zImage -j 4
```

编译成功后就会在当前目录下得到文件compile_commands.json，需要如下修改：



在gedit中使用快捷键"Ctrl+H"即可如下操作：

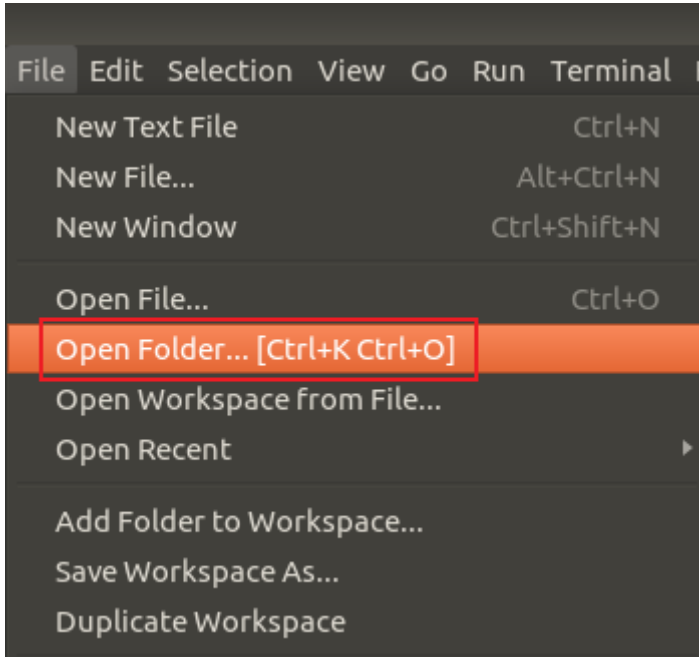


7.2 使用vscode打开内核

7.2.1 打开目录

有两种方法：

- 在vscode中入下操作，选择、打开目录"/home/book/100ask_imx6ull-sdk/Linux-4.9.88"

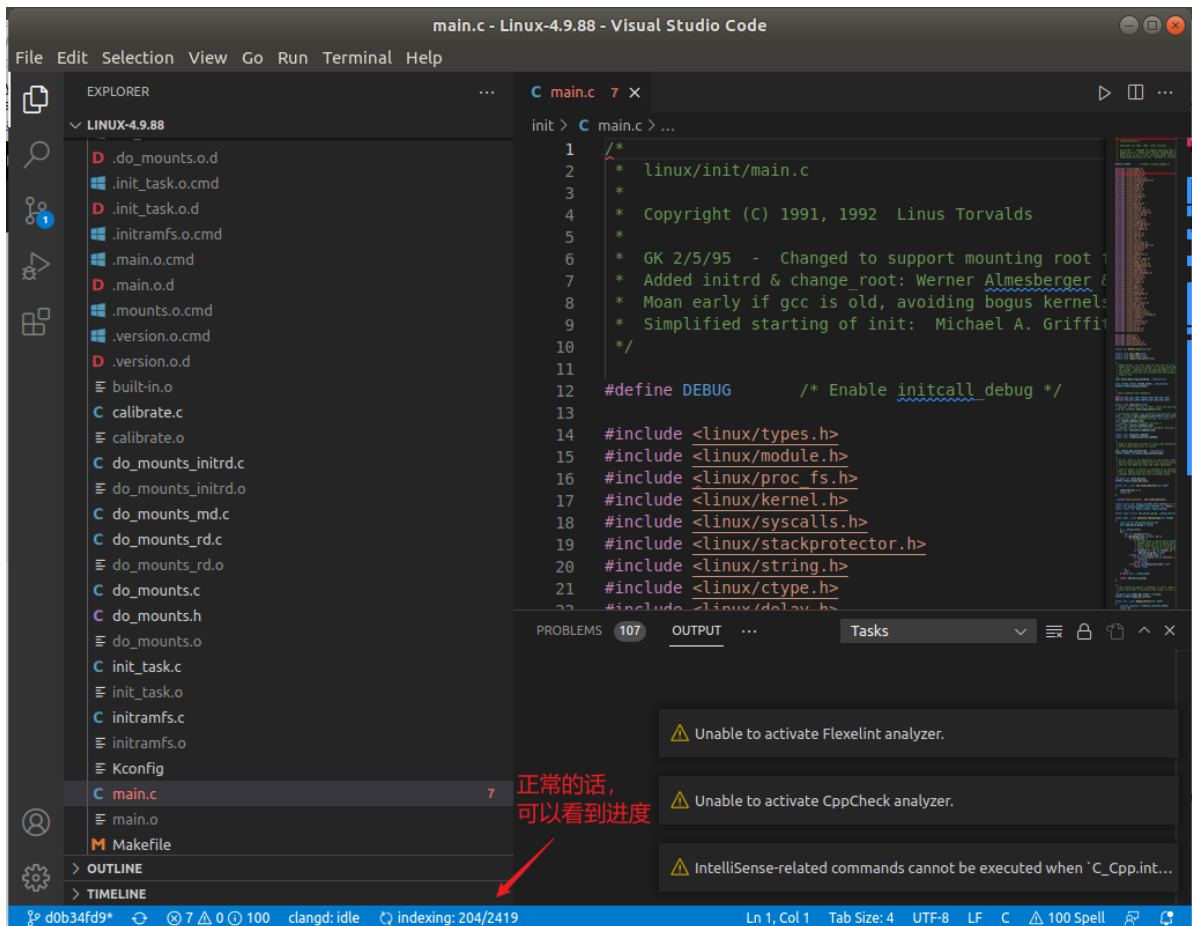


- 在终端里进入内核目录，执行命令

```
code .
```

7.2.2 触发clangd建立索引

在vscode里打开任意一个C文件，就会触发clangd建立索引：



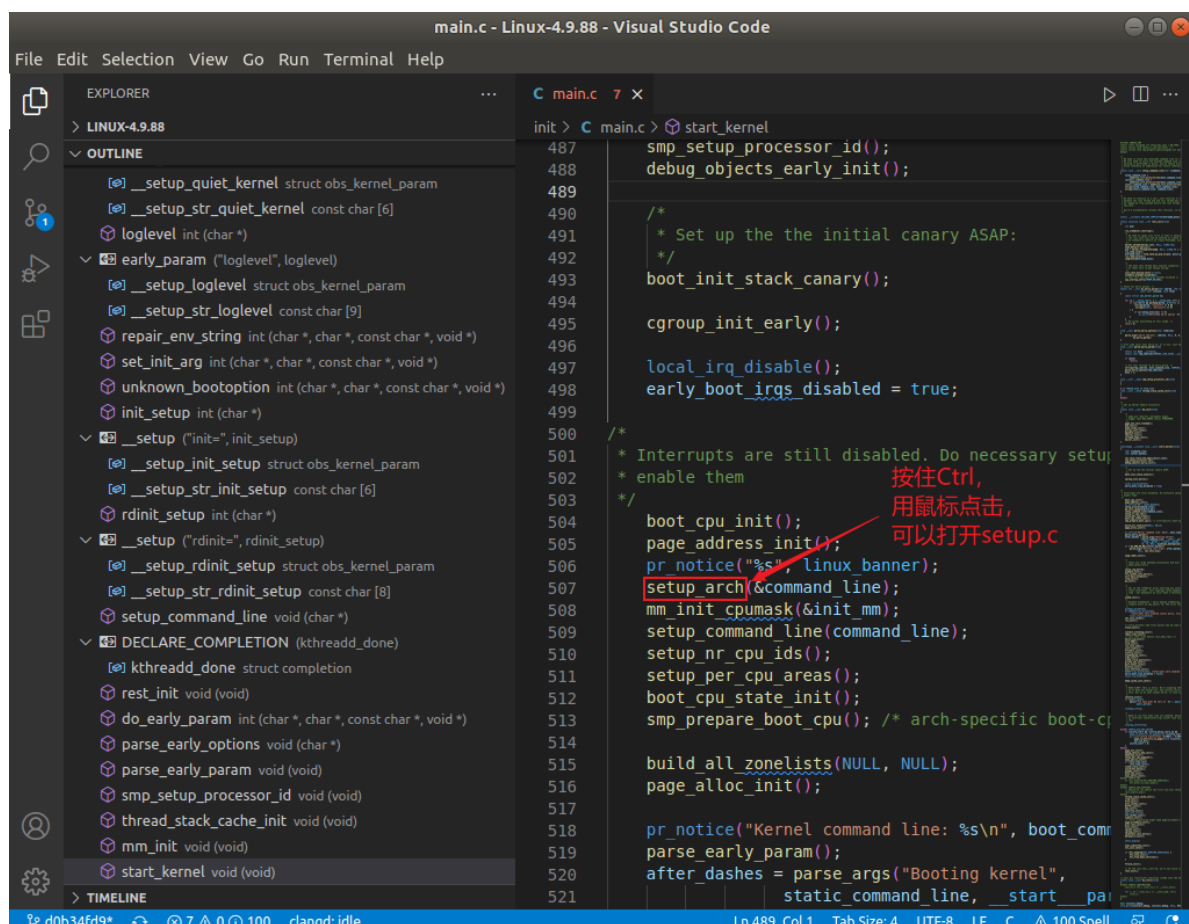
如果没有看到上述状态，可以如下处理：

- 按照《2.3.2 配置clangd》重新编辑setting.json
- 重新启动vscode、重新打开内核源码目录、重新打开C文件

在创建索引的过程中，可以使用如下命令查看.cache目录，它会不断变大(最终大小在60M左右)：

```
book@100ask: ~/100ask_imx6ull-sdk/Linux-4.9.88
File Edit View Search Terminal Help
book@100ask:~/100ask_imx6ull-sdk/Linux-4.9.88$
book@100ask:~/100ask_imx6ull-sdk/Linux-4.9.88$ du -h .cache/
68M    .cache/clangd/index
68M    .cache/clangd
68M    .cache/
book@100ask:~/100ask_imx6ull-sdk/Linux-4.9.88$
```

7.2.3 验证



8. 使用vscode阅读内核外部的源码

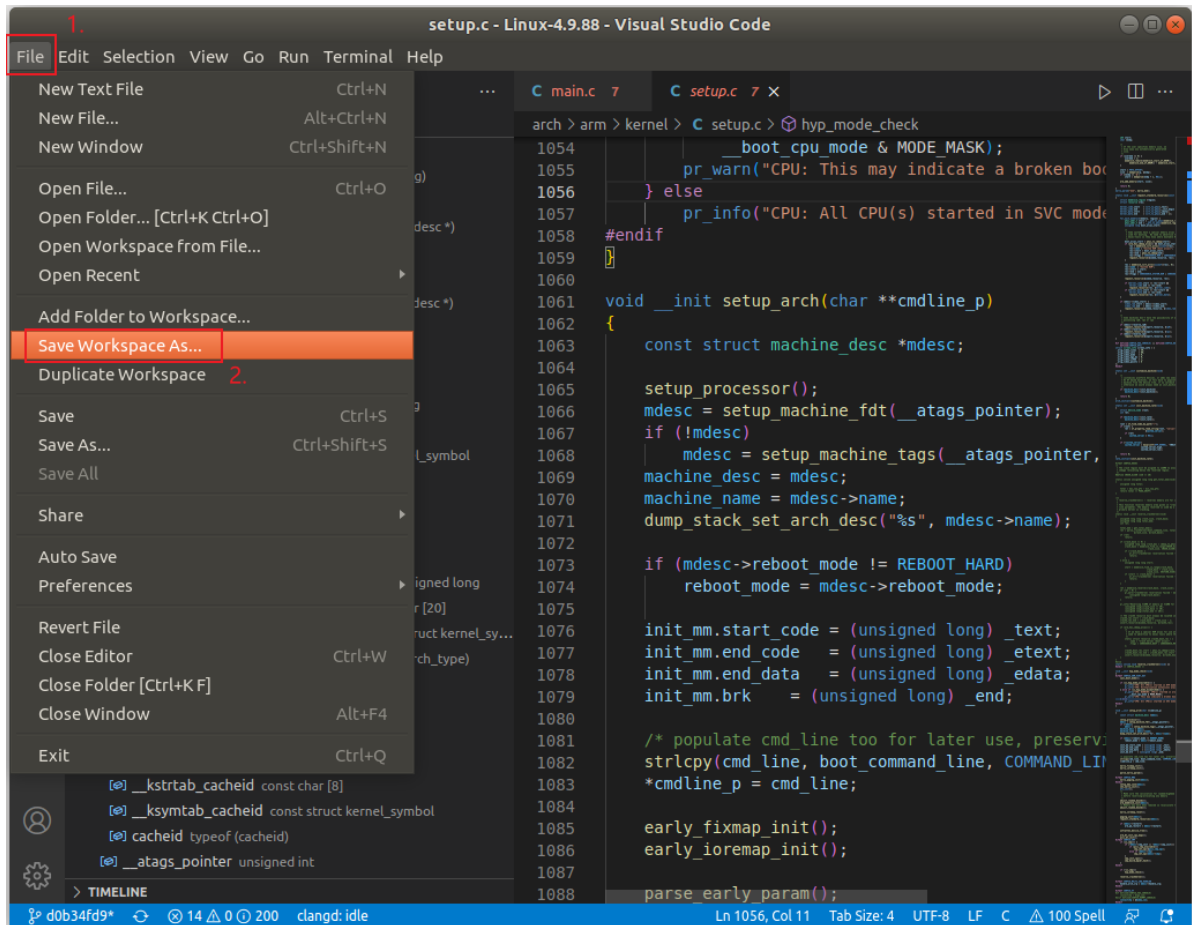
比如我们编写了hello驱动程序，它用到内核里的头文件、函数，我们点击hello驱动里的函数时，想打开内核的文件。

需要创建一个workspace：

- 里面含有内核目录、hello驱动源码目录
- 内核目录下有compile_commands.json
- hello驱动源码目录下有compile_commands.json

8.1 创建workspace

使用vscode打开内核目录，然后保存为WorkSpace，如下操作：



保存在内核源码的上一层目录里：



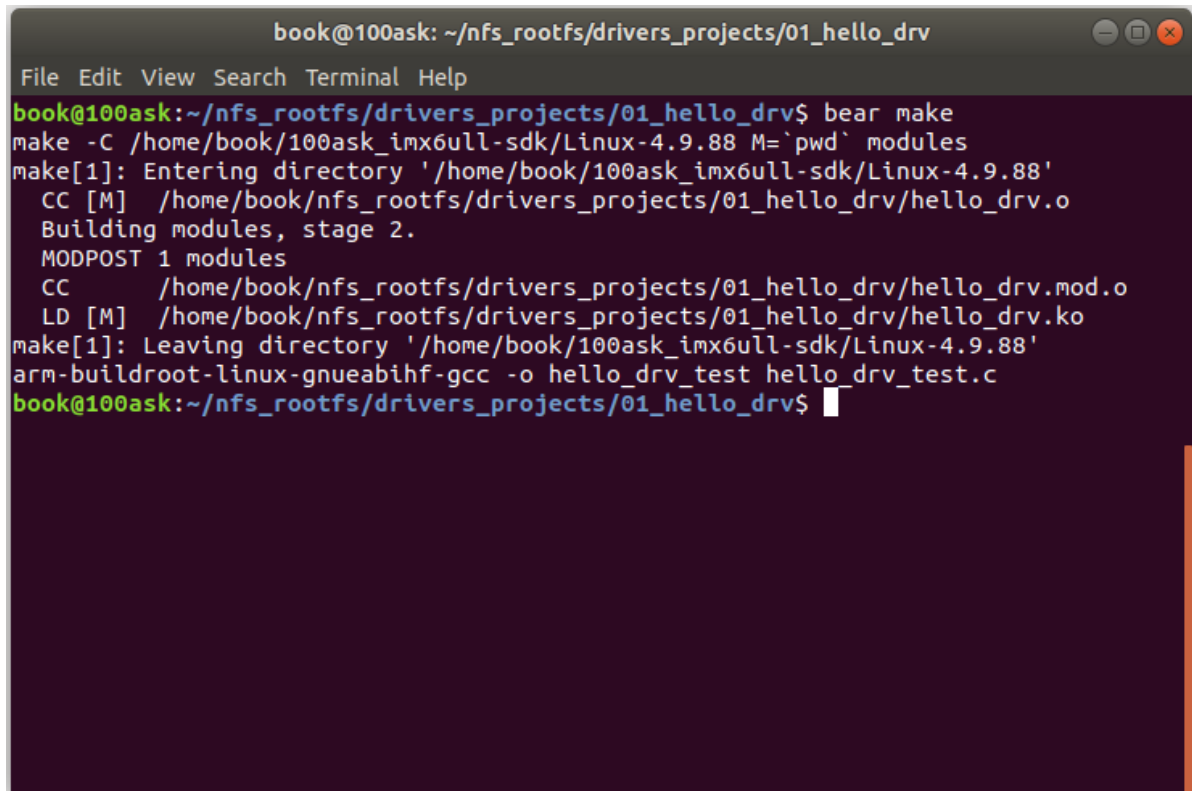
8.2 把驱动目录加入workspace

假设驱动程序位于这个目录： `/home/book/nfs_rootfs/drivers_projects/01_hello_drv/`。

8.2.1 编译驱动

使用如下命令编译，它会生成`compile_commands.json`：

```
cd /home/book/nfs_rootfs/drivers_projects/01_hello_drv/  
bear make
```

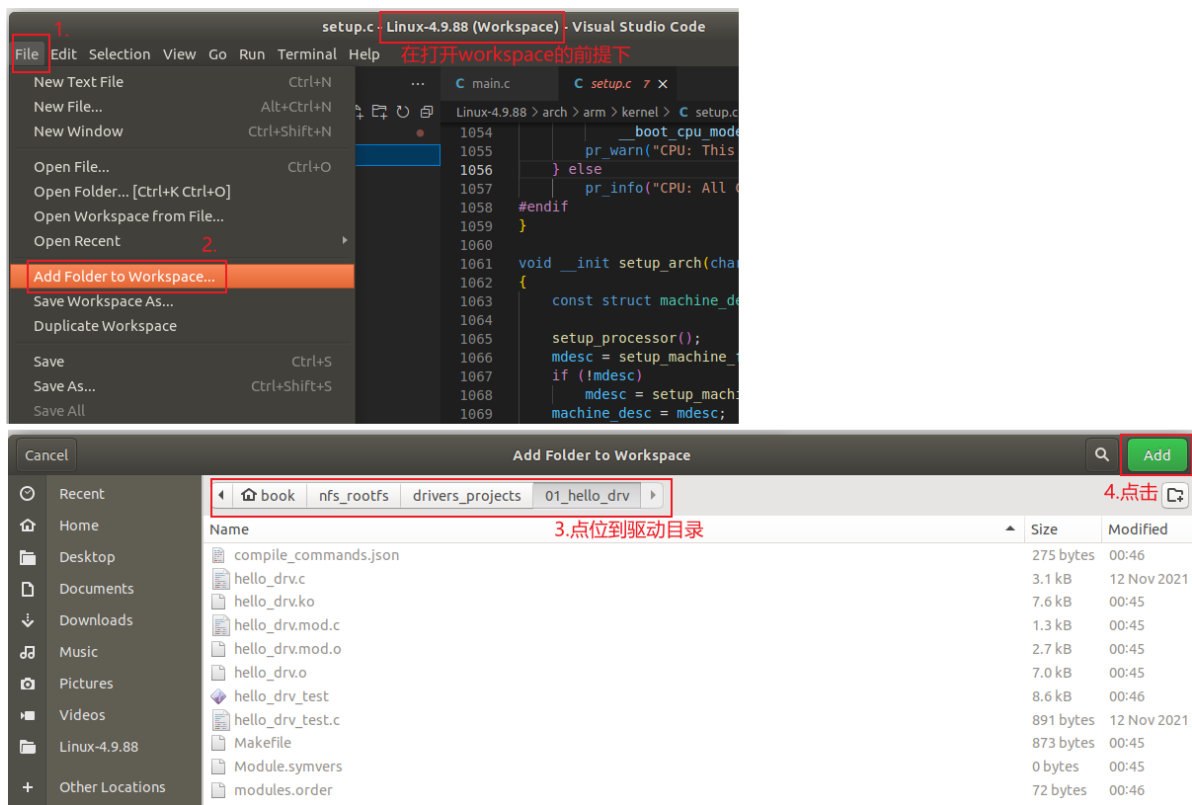


```
book@100ask: ~/nfs_rootfs/drivers_projects/01_hello_drv  
File Edit View Search Terminal Help  
book@100ask:~/nfs_rootfs/drivers_projects/01_hello_drv$ bear make  
make -C /home/book/100ask_imx6ull-sdk/Linux-4.9.88 M='pwd' modules  
make[1]: Entering directory '/home/book/100ask_imx6ull-sdk/Linux-4.9.88'  
  CC [M]  /home/book/nfs_rootfs/drivers_projects/01_hello_drv/hello_drv.o  
  Building modules, stage 2.  
  MODPOST 1 modules  
  CC      /home/book/nfs_rootfs/drivers_projects/01_hello_drv/hello_drv.mod.o  
  LD [M]  /home/book/nfs_rootfs/drivers_projects/01_hello_drv/hello_drv.ko  
make[1]: Leaving directory '/home/book/100ask_imx6ull-sdk/Linux-4.9.88'  
arm-buildroot-linux-gnueabi-hf-gcc -o hello_drv_test hello_drv_test.c  
book@100ask:~/nfs_rootfs/drivers_projects/01_hello_drv$
```

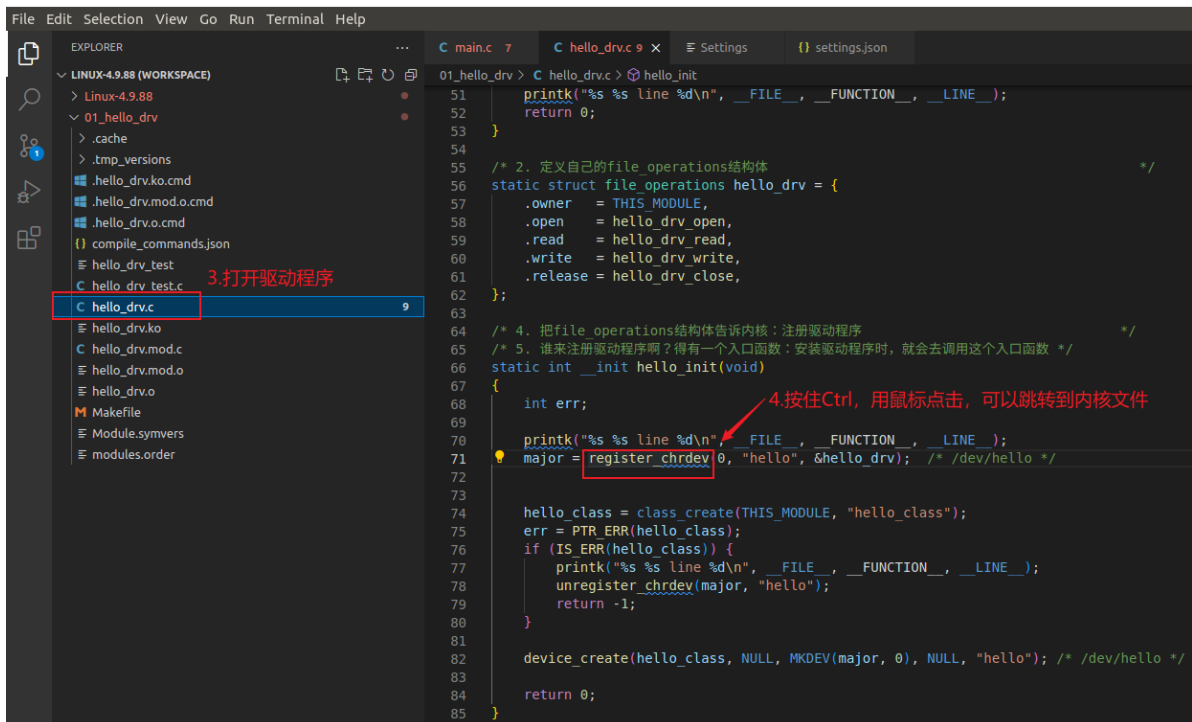
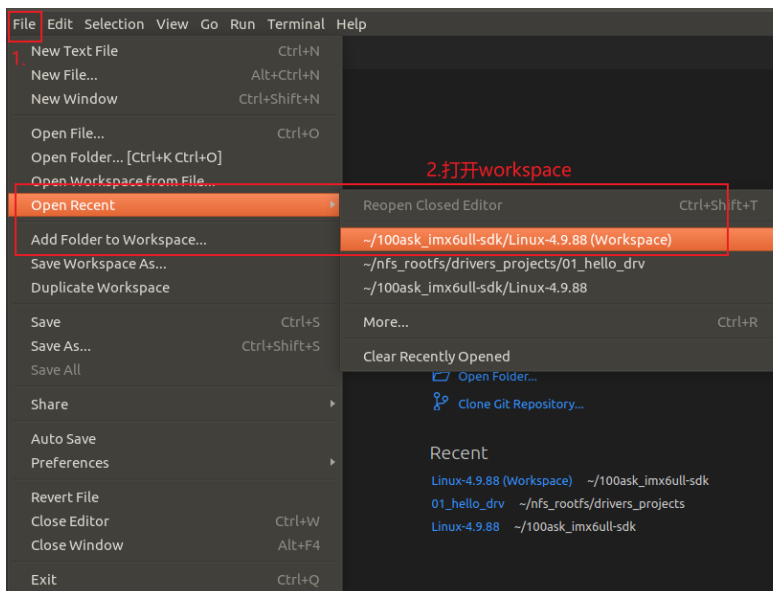
8.2.2 修改`compile_commands.json`

把里面的`"cc"`全部修改为`"arm-buildroot-linux-gnueabi-hf-gcc"`。

8.2.3 加入workspace



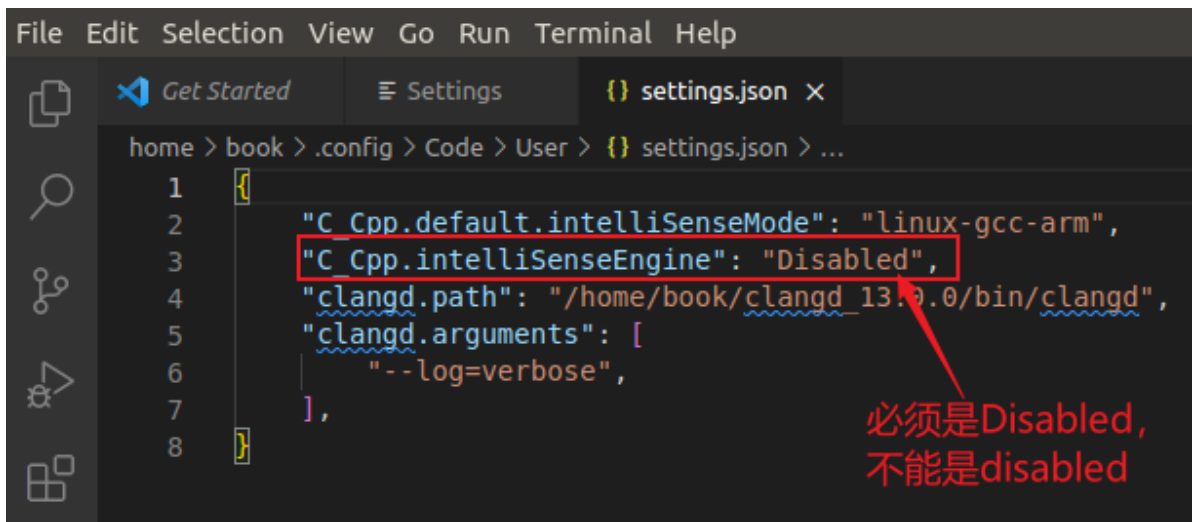
8.3 验证



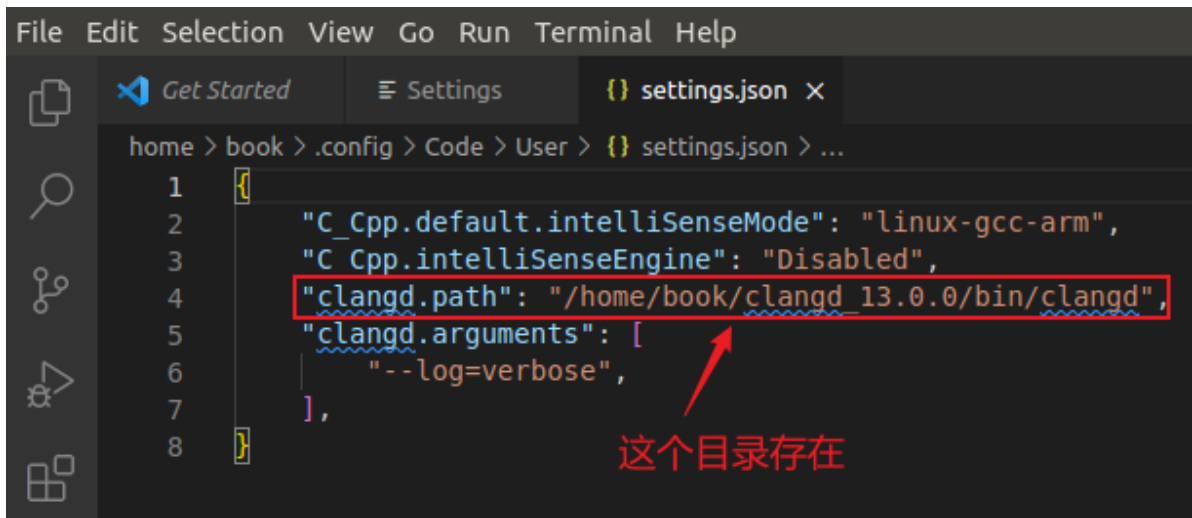
9. 常见错误

9.1 无法跳转

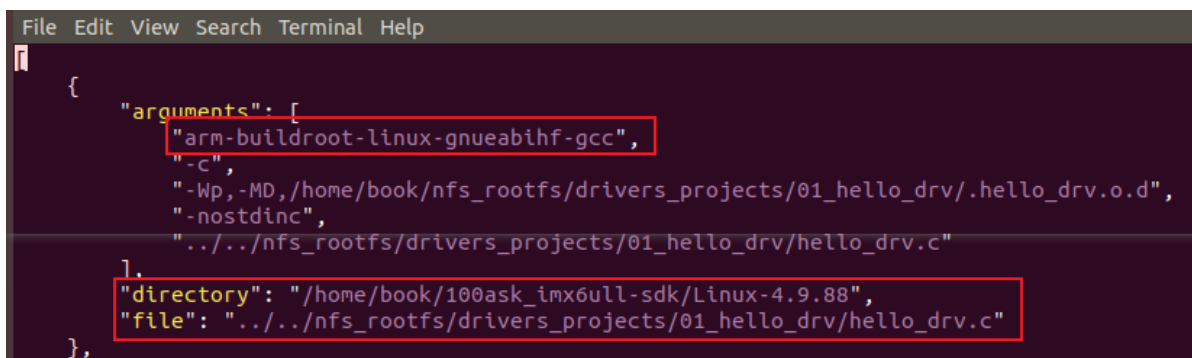
第1步，确认已经关闭intellisense：



第2步，确认有clangd：



第3步，确认源码目录下有compile_commands.json，并且文件里面记录C文件、"cc"被改成了"arm-buildroot-linux-gnueabi-hf-gcc"：



第4步，在vscode里打开C文件后，确认.cache目录生成了：

```
book@100ask:~/100ask_imx6ull-sdk/Linux-4.9.88$  
book@100ask:~/100ask_imx6ull-sdk/Linux-4.9.88$ du -h .cache/  
68M      .cache/clangd/index  
68M      .cache/clangd  
68M      .cache/  
book@100ask:~/100ask_imx6ull-sdk/Linux-4.9.88$
```

```
book@100ask:~/nfs_rootfs/drivers_projects/01_hello_drv$  
book@100ask:~/nfs_rootfs/drivers_projects/01_hello_drv$ du -h .cache/  
16K      .cache/clangd/index  
20K      .cache/clangd  
24K      .cache/  
book@100ask:~/nfs_rootfs/drivers_projects/01_hello_drv$  
book@100ask:~/nfs_rootfs/drivers_projects/01_hello_drv$
```