# 安装32位ubuntu系统，并制作成docker镜像

为了方便调试16位，32位linux0.11代码，这里选择了ubuntu32（不是x64）

## 下载安装盘

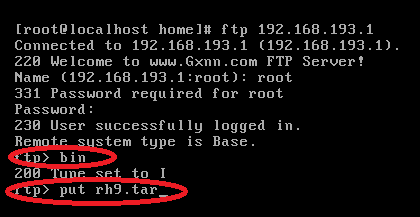
wget <http://releases.ubuntu.com/xenial/ubuntu-16.04.6-server-i386.iso>

## 安装ubuntu（不安装任何包）

## 打包系统

tar -cvf /home/ubuntu.tar / --exclude=proc --exclude=dev --exclude=run --exclude=boot --exclude=/home --exclude=/root --exclude=/opt --exclude=/mnt --exclude=/sys

通过ftp将ubuntu.tar放到主机上



## 导入docker镜像

cat ubuntu.tar | docekr import - ubuntu.16.i386

# 创建32位的ubuntu的docker容器

为了方便调试16位，32位linux0.11代码，这里选择了ubuntu32（不是x64）

在主机上运行

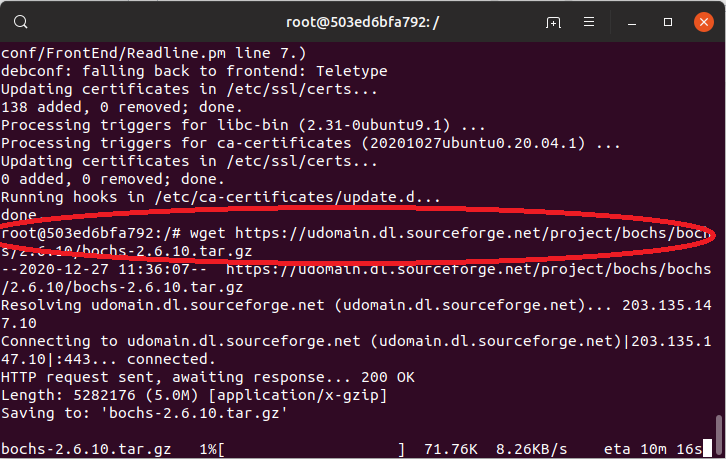
apt-get install x11-xserver-utils

xhost +

docker run -it -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY -v /*yy:/*yy ubuntu.16.i386 /*bin/bash*

# 在docker容器里下载并编译bochs

wget [https://sourceforge.net/projects/bochs/files/bochs/2.6.10/bochs-2.6.10.tar.gz](https://sourceforge.net/projects/bochs/files/bochs/2.6.10/bochs-2.6.10.tar.gz/download)



tar -zxvf bochs-2.6.10.tar.gz

cd bochs-2.6.10

*apt install gcc g++ gtk2.0-dev xorg-dev build-essential*

./configure --enable-gdb-stub

make

make install

# 在docker容器里使用bochs运行linux0.11

### 下载linux0.11的运行文件

wget <http://www.oldlinux.org/Linux.old/bochs/linux-0.11-gdb-rh9-050619.tar.gz>

tar -zxvf linux-0.11-gdb-rh9-050619.tar.gz

cp -fR ./linux-0.11-gdb-rh9-050619/linux/ /yy/linux/

cd /yy/linux

### 通过vi新建文件1.bxrc

vi 1.bxrc

并输入下面4行

gdbstub: enabled=1, port=1234, text\_base=0, data\_base=0, bss\_base=0

floppya: 1\_44="bootimage-0.11-hd", status=inserted

ata0-master: type=disk, path="hdc-0.11-new.img", mode=flat, cylinders=410, heads=16, spt=38

boot: a

### 通过vi新建文件2.sh

vi 2.sh

输入下面文字

bochs -q -f 1.bxrc

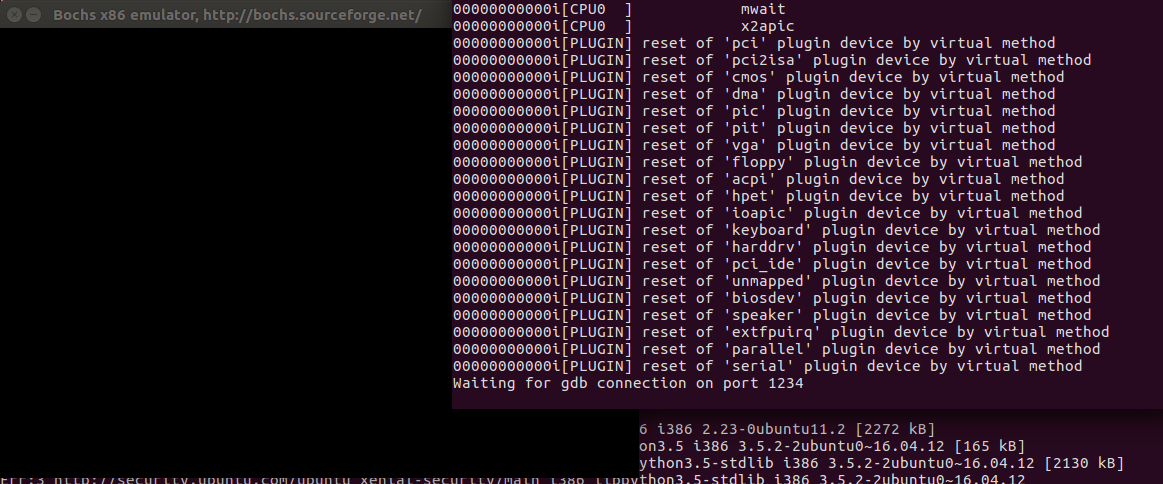
### 修改2.sh的权限为：可执行

chmod 777 2.sh

### 执行2.sh

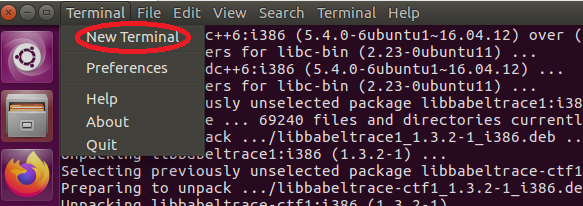
./2.sh

等待vscode连接



# 在docker容器里安装vscode

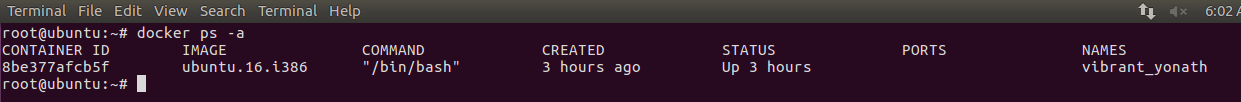
## 创建新的终端



## 再次登录到上面的容器

1. 找到正在运行中的容器

docker ps -a



1. 登录到容器中

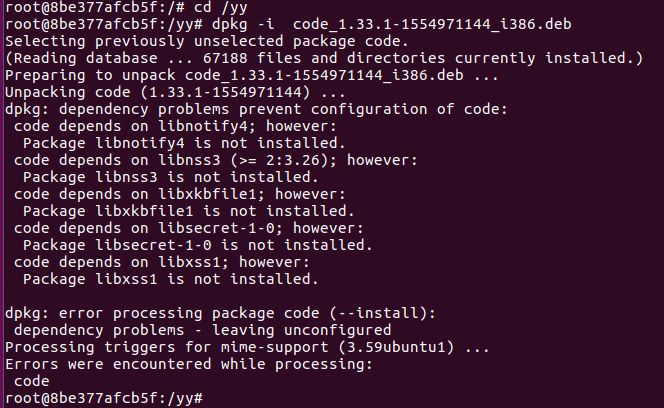
docker exec -it 8be377afcb5f /bin/bash

## 下载vscode安装文件

wget https://az764295.vo.msecnd.net/stable/ea3859d4ba2f3e577a159bc91e3074c5d85c0523/code\_1.33.1-1554971144\_i386.deb

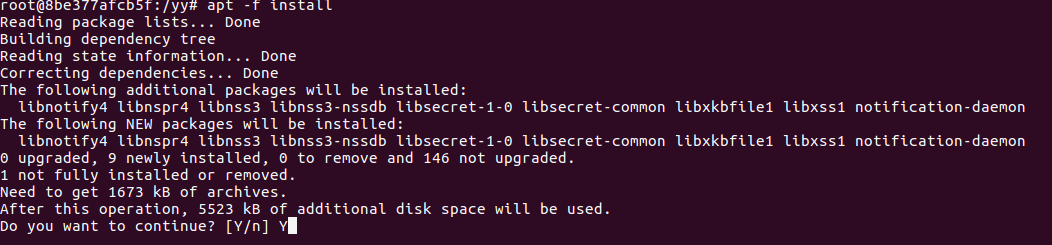
## 安装vscode

1. dpkg -i code\_1.33.1-1554971144\_i386.deb



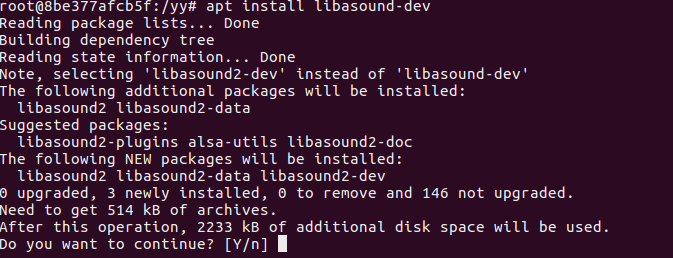
1. 强制安装

apt -f install



1. 安装libasound

apt install libasound-dev



1. 安装gdb

apt install gdb

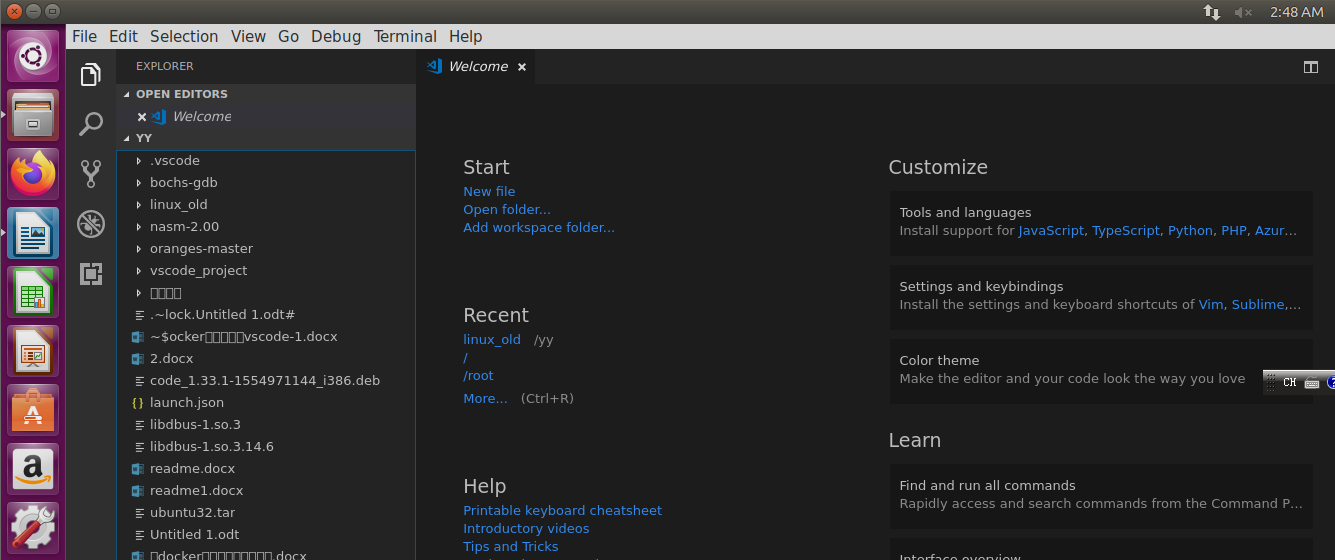
1. 运行vscode

sudo code . --user-data-dir='/yy/vscode\_project'

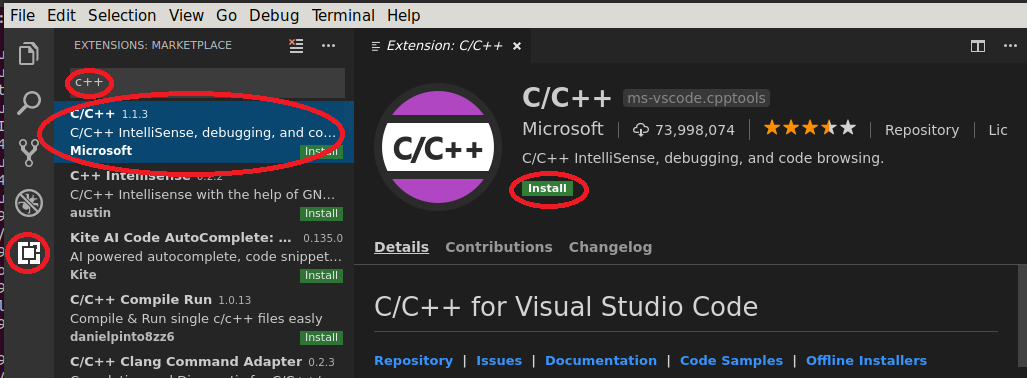


没有报任何错误

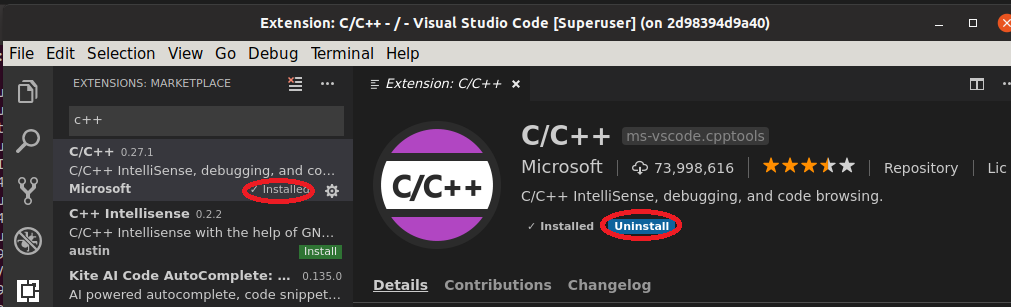
过一会儿，会出现vscode的窗口



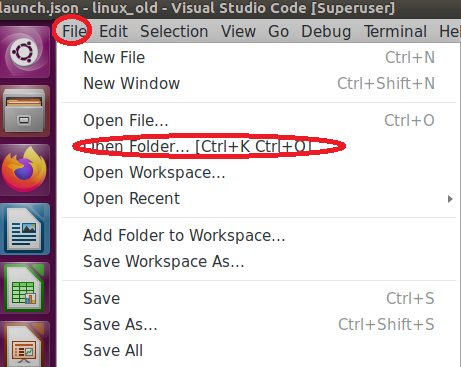
## 在vscode安装c/c++插件

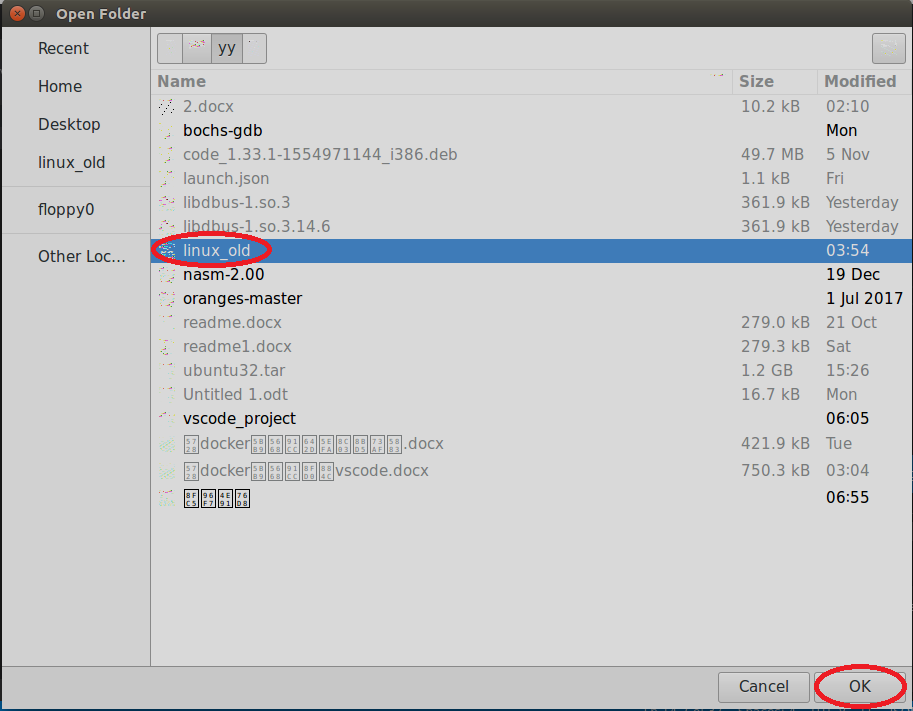


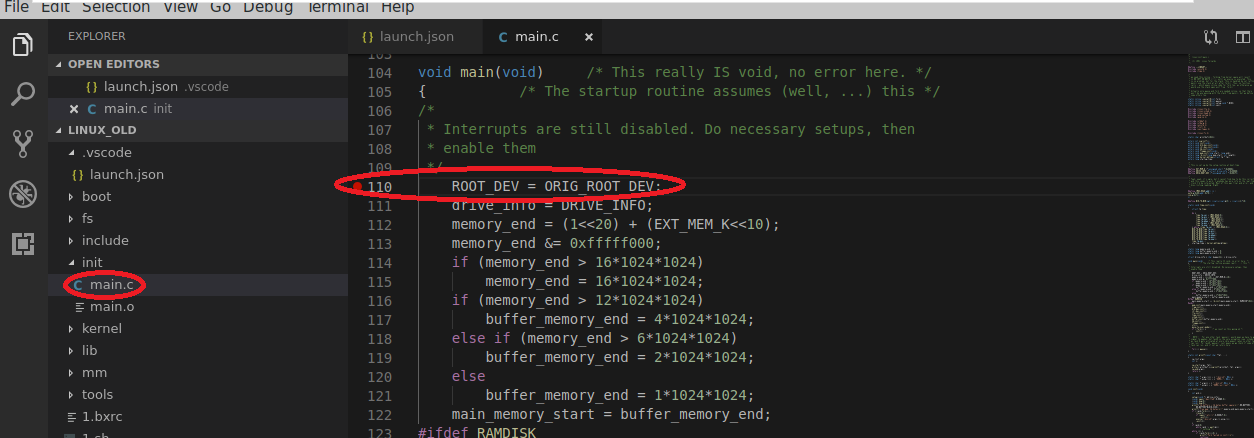
安装完成后，插件状态发生变化（如下图，红色圈中的文字）



## 打开linux0.11源代码目录,按F9设置断点

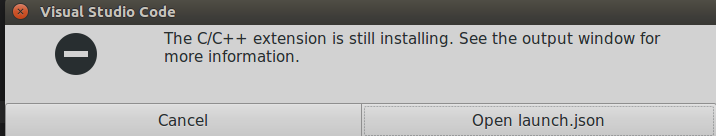






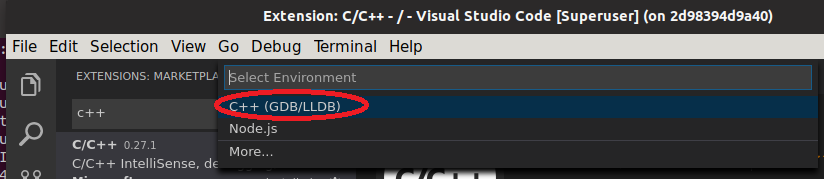
将光标移动到指定行， 按F9设置或取消断点

## 按F5运行



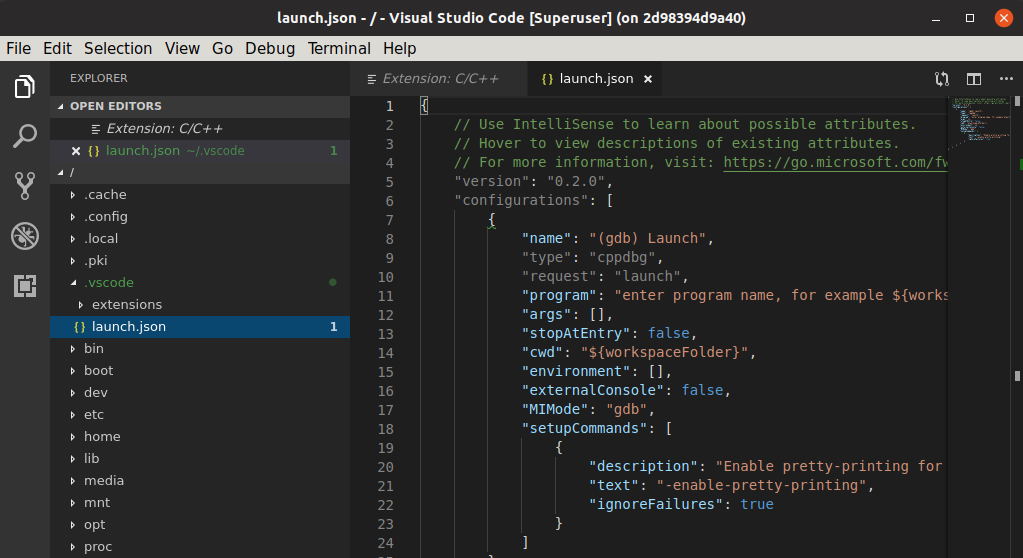
等会儿，等安装完成再按F5重新运行，

Vscode弹出窗口，让选择调试器，这里选择“c++(GDB/LLDB)”



## 设置launch.json配置文件

选好调试器后，vscode会自动创建出launch.json, 但是这是配置的c++本地启动。调试linux0.11需要的是远程调试， 所以使用下面的文字替换掉整个文件



{

"version": "0.2.0",

"configurations": [

{

"name": "gdb Remote Launch",

"type": "cppdbg",

"request": "launch",

"program": "/yy/linux\_old/tools/system",

"stopAtEntry": true,

"environment": [],

"externalConsole": false,

"MIMode": "gdb",

"miDebuggerPath": "gdb",

"miDebuggerArgs": "gdb",

"linux": {

"MIMode": "gdb",

"miDebuggerPath": "/usr/bin/gdb",

"miDebuggerServerAddress": "127.0.0.1:1234",

},

"logging": {

"moduleLoad": false,

"engineLogging": false,

"trace": false

},

"setupCommands": [

{

"description": "Enable pretty-printing for gdb",

"text": "-enable-pretty-printing",

"ignoreFailures": true

}

],

"cwd": "${workspaceFolder}",

}

]

}

## 再按F5开始调试， 程序运行到断点暂停等待调试指令

## 