# 准备一个32位的linux的机器，安装好docker

# 在主机创建一个目录

mkdir /yy1

# 运行编译的容器

1. 启动容器

docker run -itv /yy1:/yy1 yy20190811/rh9-to-compile-linux0.11-core

后面的命令都在容器中执行

1. 将源代码拷贝到/yy1/下

cp -fR /yy/linux\_old/ /yy1/

1. 从重新编译linux0.11的源代码

cd /yy1/linux\_old

make clean && make

1. 重新制作安装盘

./1.sh

# 运行调试环境

## 运行docker容器

1. 在主机新启动一个console，后面命令在新的console中执行
2. 在主机安装软件

apt-get install x11-xserver-utils

xhost +

1. 启动容器

docker run -it -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY -v /yy1:/yy1 yy20190811/ubuntu-2-debug-linux0.11 /bin/bash

## 启动bochs

在ubuntu-2-debug-linux0.11-core容器中

### 进入yy1的linux\_old目录

cd /yy1/linux\_old

### 启动bochs

./2.sh

bochs等待

## 启动vscode

### 在主机新启动一个新的console，找到上步启动的docker容器的id

docker ps -a

例如： 查询到容器的id是：8be377afcb5f

后面的命令都在这个新console中执行

### 进入到docker容器中

docker exec -it 8be377afcb5f /bin/bash

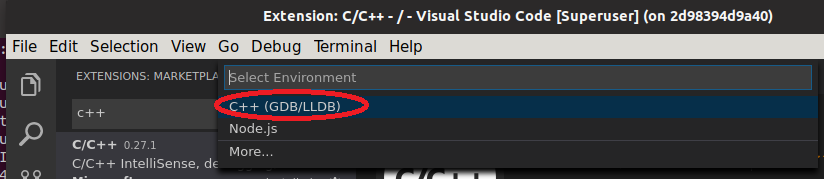
### 启动vscode

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

### vscode打开新的目录(/yy1/linux\_old)

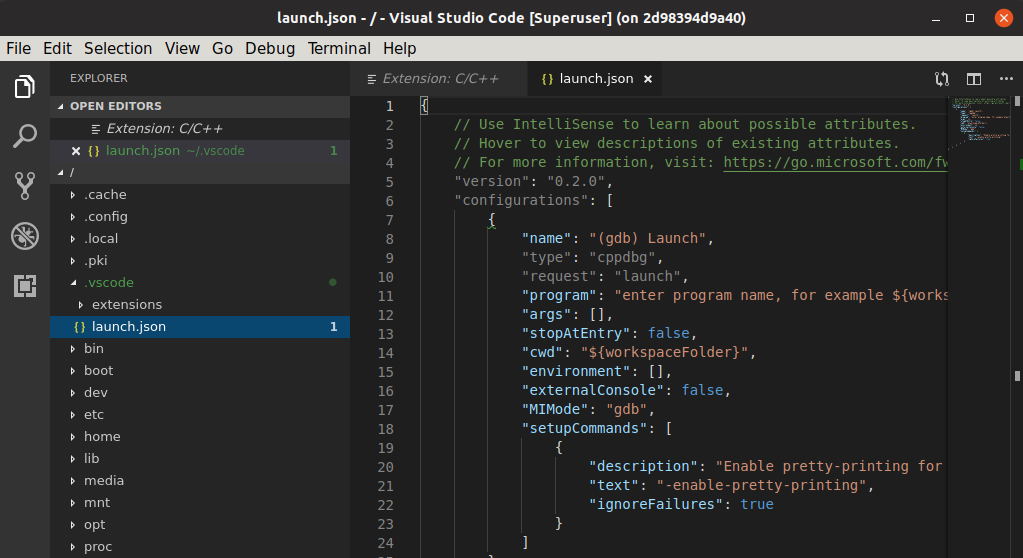
### 按F5

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": "/yy1/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

## 