## 环境

本地环境：win8

远程环境：ubuntu-16.04（使用VirtualBox创建的虚拟机）

## 工具

本地环境：

Visual Studio Code（版本1.52.1）

Git（版本2.25.0.windows.1）

远程环境：

openssh（ubuntu系统自带）

gcc

ubuntu系统会自带gcc，如果没有可以使用如下命令安装

更新包列表

sudo apt-get update

安装gcc

sudo apt-get install build-essential

gdb

安装命令：sudo apt-get install gdb

## 安装Remote-SSH插件

安装Remote-SSH插件，该插件通过ssh连接远程服务器。

安装Remote Development插件也可以，安装该插件将会同时安装Remote-Container, Remote-SSH 及 Remote-WSL这三个插件。

点击【扩展】，然后搜索remote ssh，点击【安装】即可。

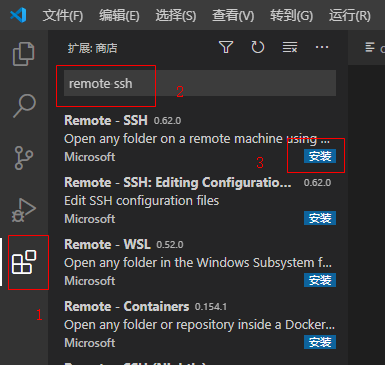


图3-1

安装完成后，可以在左边的活动栏中发现多了一个按钮：远程资源

管理器。

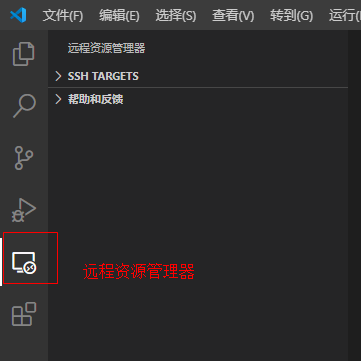


图3-2

## 安装SSH客户端

Remote-SSH插件需要借助本地的ssh客户端进行连接服务器，所以需要在本地安装ssh客户端。

官网给出的安装方法，如图所示：

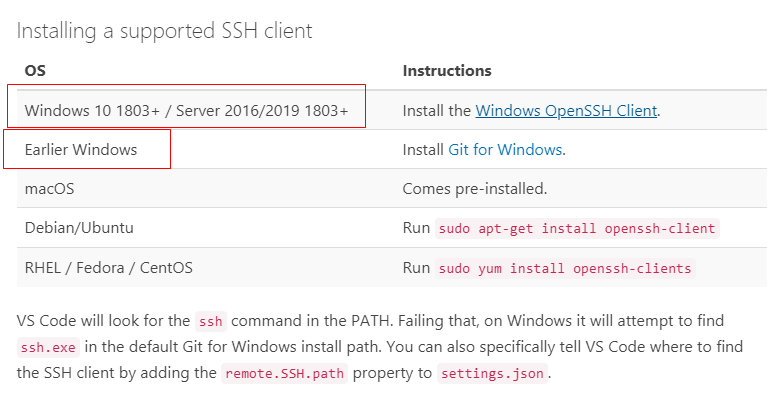


图5-1

可以从上图看到，win10可以直接安装ssh客户端，而非win10则可以安装Git。

如果是win10操作系统可以在powershell中直接安装，命令如下：

Get-WindowsCapability -Online | ? Name -like 'OpenSSH\*'

Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0

如果非win10操作系统，可以安装Git，Git自带的工具中包含ssh客户端。

安装Git后，可以在Git的安装目录中找到ssh.exe程序，该程序就是ssh客户端。

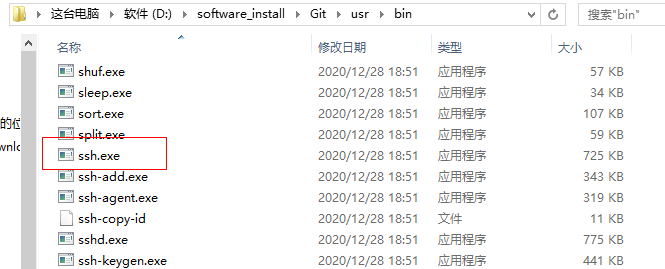


图5-2

如上图所示，在Git的user/bin目录中可以看到ssh.exe

## 配置ssh客户端路径

为了让Remote-SSH插件找到ssh客户端，需要将ssh客户端的路径配置到插件中。

打开设置



图5-3

在设置中找到【用户->扩展->RemoteSSH】

（1）修改RemoteSSH:path配置，将ssh.exe的完整路径填进去。我本地的ssh.exe客户端的路径是D:\software\_install\Git\usr\bin\ssh.exe，所以图中所示填的就是该路径。这样Remote-SSH就可以使用ssh.exe作为ssh客户端连接服务器。

（2）在RemoteSSH:Show Login Teminal配置上打上对勾.该项是配置每次登录时都打开终端。

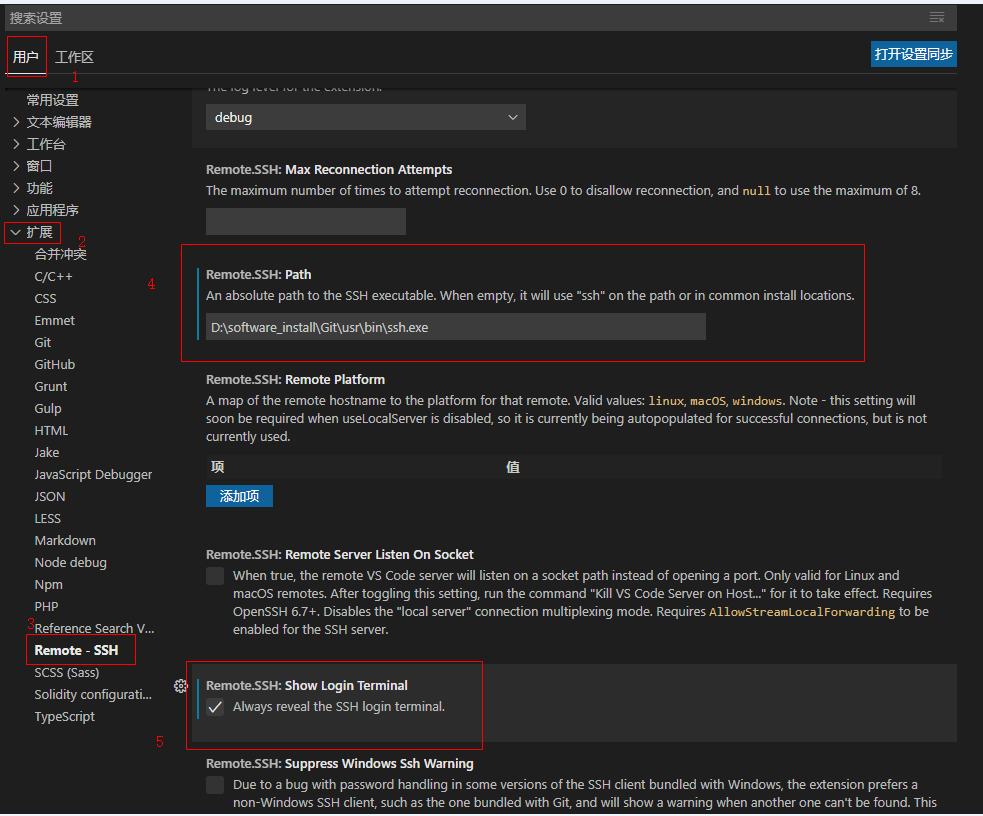
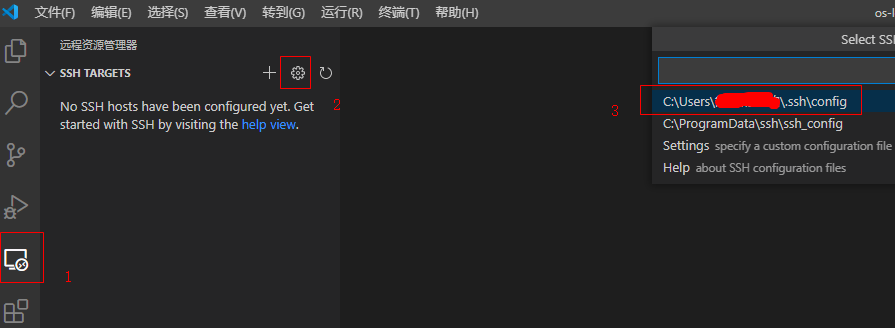


图5-4

配置登录时打开终端

## 配置服务器的地址、端口号和登录用户

依次点击 远程资源管理器->配置，然后弹出如图3处所示的选择框。这个选择框，是选择config配置文件存放的路径。选择第一个就可以，存放在当前用户的.ssh目录下。该配置文件用于记录服务器的IP地址、端口号和用户名。



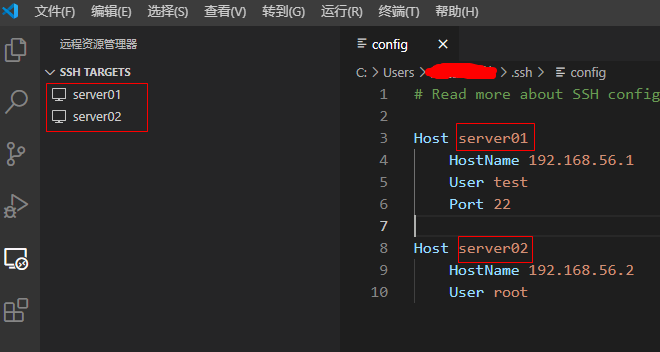
我的config配置如图所示。

Host是服务器名称，可以随便写。

HostName是服务器IP

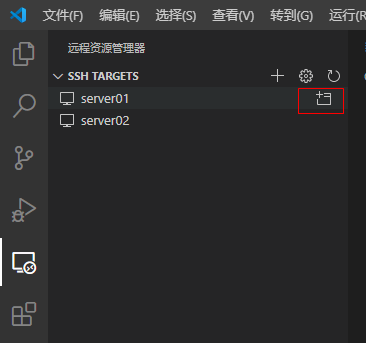
Port是服务器openssh服务监听的端口号，一般都是22,。该项也可以不填，不填则默认为22.

在config中可以配置多个服务器，配置的服务器会在左侧显示。

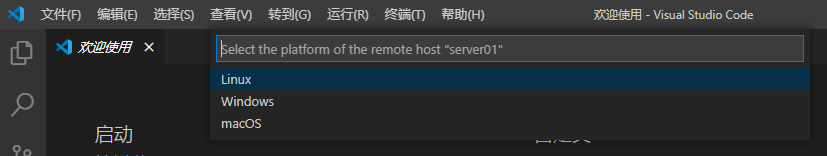


## 连接远服务器

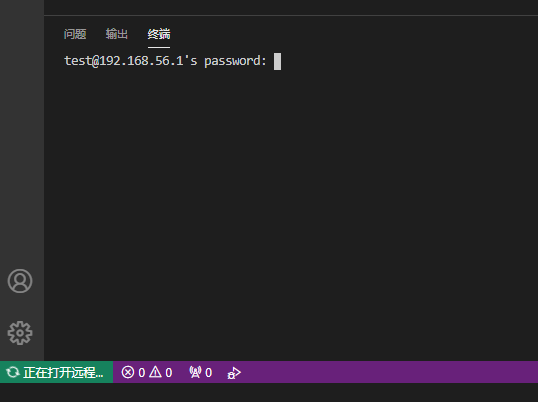
点击 server01那一栏后面的按钮就可以在新的窗口连接server01。



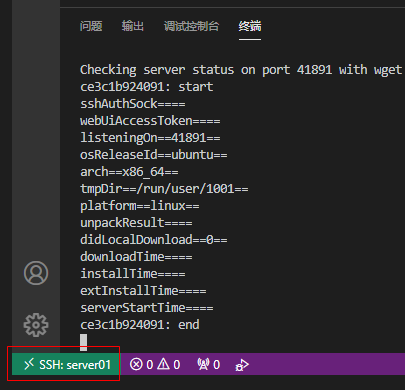
点击后，生成新的窗口，并弹出选择框，选择服务器的操作系统。我的服务器是ubuntu，所以选择Linux。



在新窗口的终端里输入所登录用户的密码。



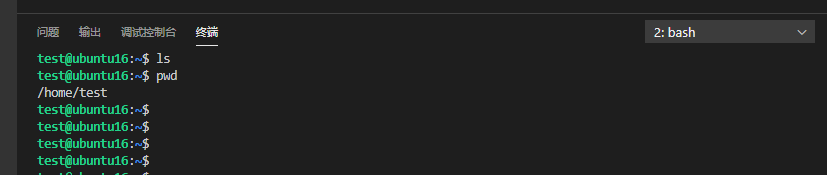
输入密码后，等待建立连接，连接成功后，左下角的绿色区域显示SSH:server01，代表连接成功。



此时，可以新建一个终端，操作服务器。依次点击【终端】->【新终端】，就会建立一个新终端。



新的终端如下图所示，可以远程操作服务器了。



## 选择一个远程目录作为工作区

上面的操作是使用ssh连接服务器，接下来就要开始远程调试程序了。

点击【资源管理器】，然后可以看到显示“已连接到远程”。

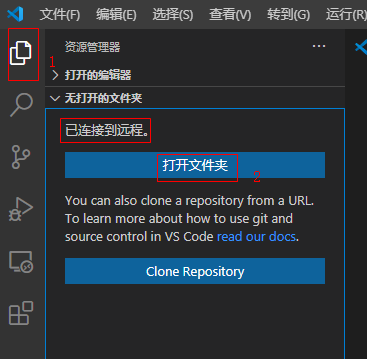


图8-1

再点击【打开文件夹】，然后就选择需要打开的文件夹。我选择/home/test/123文件夹，点击【确定】。

之后，会重新生成窗口，并需要再一次输入密码。

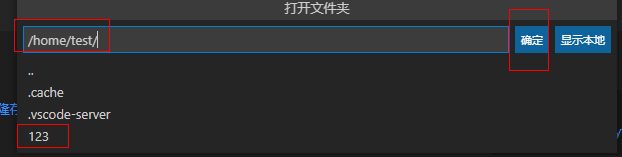


图8-2

文件夹打开后，点击图中2处的按钮【新建文件】。在这里我们新建一个hello.c文件。



图8-3

hello.c内容如下：

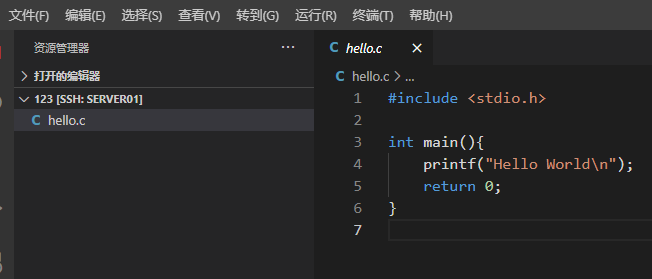


图8-4

可能此时，显示找不到头文件<stdio.h>，我们可以在c\_cpp\_properties.json文件配置includePath指定头文件的路径。

依次点击【管理】->【命令面板】



图8-5

在弹出的命令面板中搜索c/c++，点击【C/C++:编辑配置(JSON)】

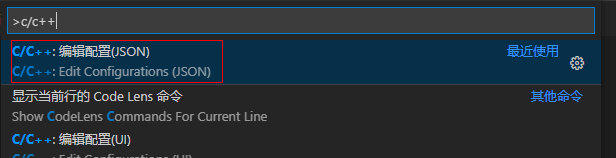


图8-6

，然后会在当前工作目录/home/test/123下创建.vscode目录，并生成c\_cpp\_properties.json文件。如下图所示：

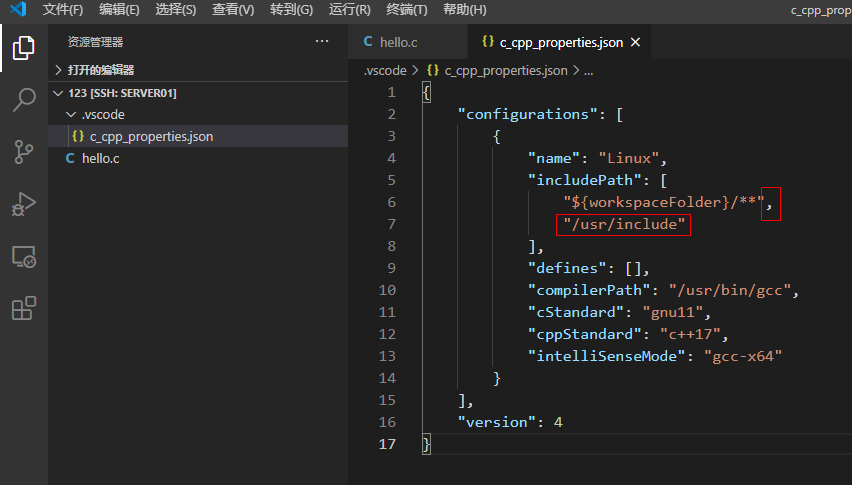


图8-7

在includePath字段中添加“/usr/include”。ubuntu系统中，stdio.h在该目录下。

## 在远程服务器上安装插件

为了可以调试，我们需要在服务器上安装插件。

点击【扩展】，然后在搜索框输入c++。可以看到C/C++和C++ Intellisense插件，需要在远程服务器上安装这两个插件。我win8本地已经安装了C/C++插件，所以该插件是灰色显示，而且安装按钮显示“在SSH:server01中安装”，点击该按钮将在服务器上安装该插件。C++ Intellisense插件我本地没有安装，所以显示白色，而且按钮是“安装”，点击该按钮将在本地和服务器上都安装该插件。

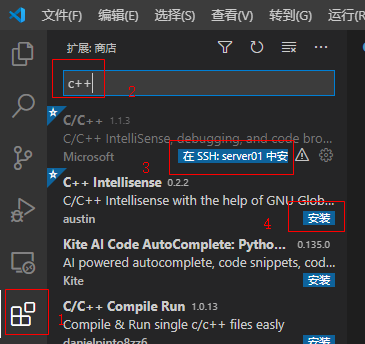


图9-1

清空搜索框，看到如下所示本地和远程的插件安装情况。可以看到，本地和远程我们都安装了这两个插件。

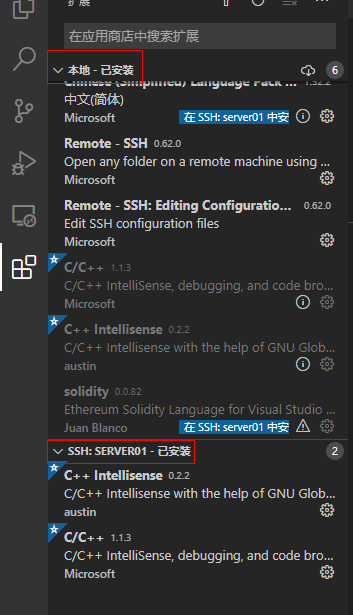


图9-2

## 创建和修改launch.json文件

点击【运行】按钮，再点击【创建launch.json文件】。

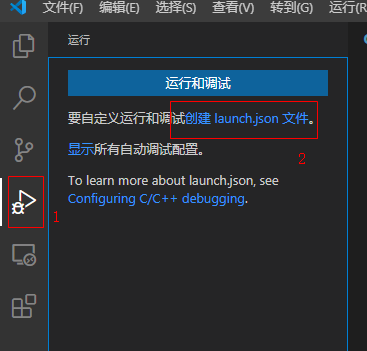


图10-1

之后，会弹出对话框，选择C++(GDB/LLDB)环境

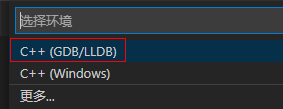


图10-2

有时候选择生成哪种配置，我们选择【默认配置】，选择其它两种配置最简单，什么都不用修改就可以调试了。但是为了学习，还是选择默认配置吧，自己去修改。

有时候不会弹出选择框，直接生成默认配置的launch.json。

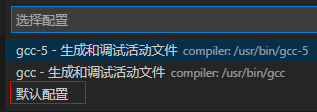


图10-3

之后，便会在当前工作目录/home/test/123下创建.vscode目录，并生成launch.json文件。如下图所示：

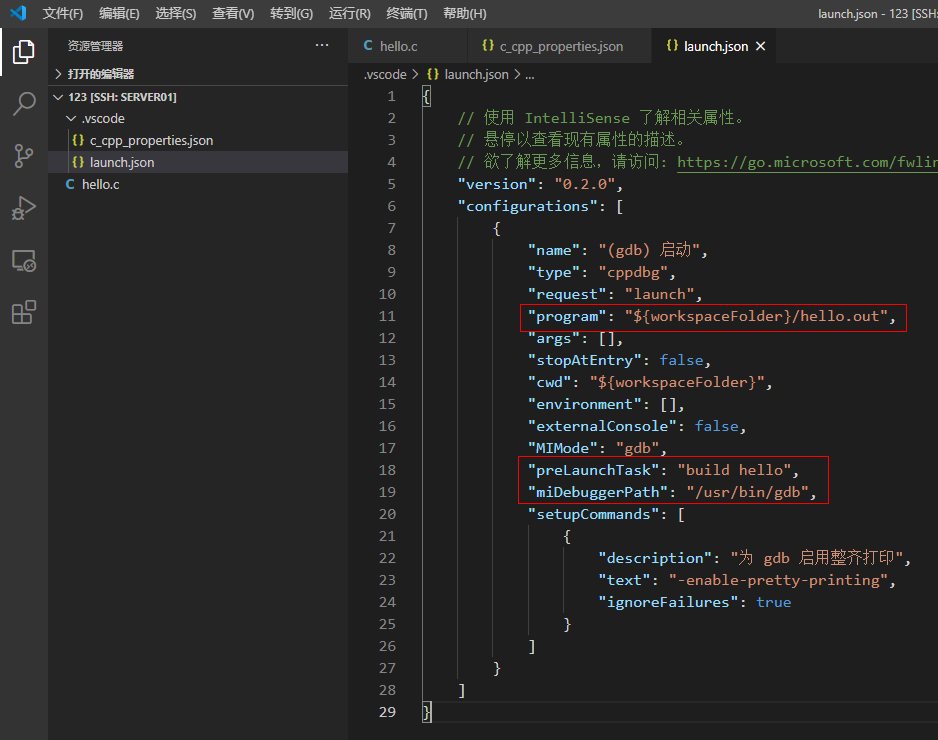


图10-4

共需要修改三个字段：

program字段指定需要运行和调试的二进制文件，也就是gcc编译hello.c生成的可执行文件hello.out。

preLaucchTask字段指定在运行和调试之前要执行的任务，一般该任务用于编译代码生成可执行文件。我们这里修改为“build hello”，因为等会我们创建任务时，将任务名取为“build hello”。

miDebuggePath字段指定调试器。

## 创建和修改task.json文件

task.json文件用于指定任务，该任务可以配置在launch.json文件中，处理调试之前需要做的准备工作。我们可以创建编译程序生成可执行文件的任务。

依次点击【终端】->【配置任务】

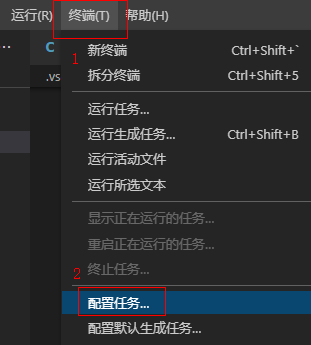


图11-1

选择【使用模板创建task.json文件】

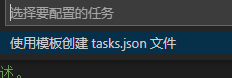


图11-2

选择【Others】

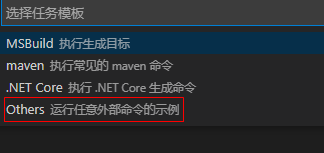
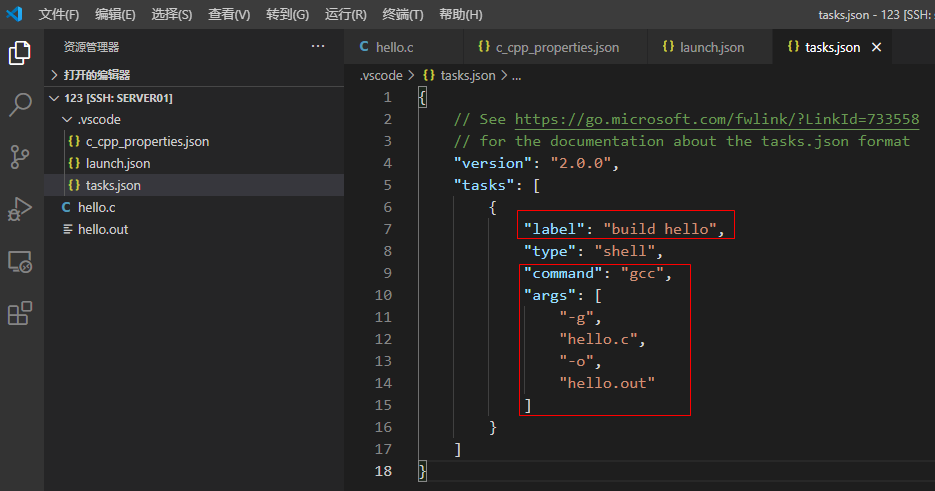


图11-3

就会在/home/test/123/.vscode中生成task.json文件



type字段是task的名称，我们在launch.json中已经指定了任务名“build hello”，所以在这里使用“build hello”作为task的名称。

command字段指定gcc命令

args字段指定命令需要的参数。

那么该task就是在shell中执行 gcc -g hello.c -o hello.out命令，编译hello.c文件生成hello.out用于运行和调试。

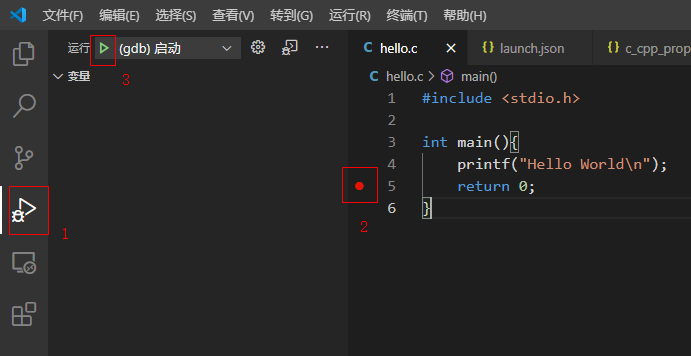
## 编写并调试程序

回到刚刚建立的hello.c文件，在某一行添加一个断点。添加断点：在某一行的前面单击一下，出现一个小红点，就代表在该行加断点。

然后点击图片中1处的【运行】，

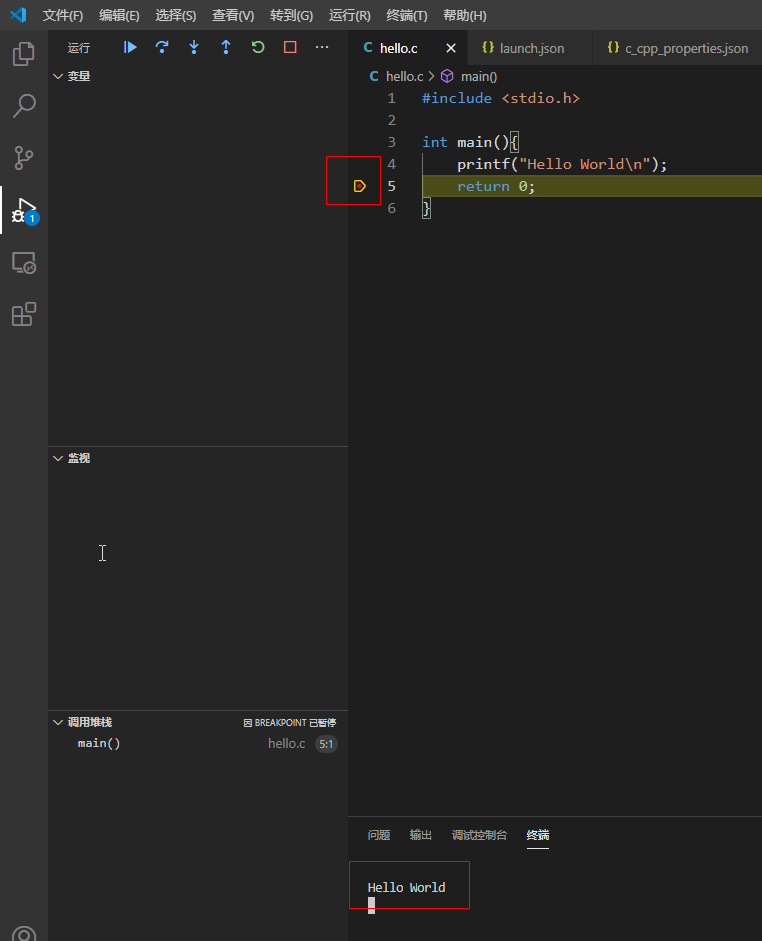
再第5行加个断点

再点击图片3处的绿色按钮【开始调试】，就可以调试了。



题12-1

调试结果如下：



## 后记

我看有的博客说需要通过gdbserver调试，我没有安装gdbserver依然可以远程调试。不知道是不是我使用最新版的VSCode，所以不需要了。

## 参考资料

1. [VSCode官网使用Remote-SSH教程](https://code.visualstudio.com/docs/remote/ssh#_getting-started)
2. [VSCode官网安装ssh客户端教程](https://code.visualstudio.com/docs/remote/troubleshooting#_installing-a-supported-ssh-client)
3. [VSCode远程开发调试服务器c/c++代码](https://blog.csdn.net/lucky_ricky/article/details/104611125)
4. [2019 VS Code 远程开发配置（热乎的）](https://blog.csdn.net/yh0503/article/details/89851899)
5. [vscode远程调试c/c++程序](https://blog.csdn.net/qq_32147775/article/details/103459473)
6. [linux系统下如何在vscode中调试C++代码](https://www.cnblogs.com/jiaxblog/p/9902535.html)
7. [vscode 远程linux调试配置](https://blog.csdn.net/u013078823/article/details/92799971)
8. [VScode Remote 远程开发与调试](https://www.jianshu.com/p/0f2fb935a9a1)
9. [VS Code Remote 发布！开启远程开发新时代](https://www.oschina.net/news/106412/vscode-remote-development?p=1)
10. [（B站视频）VS Code入门教程2020 #21 调试界面概览](https://www.bilibili.com/video/BV18E41147NM?from=search&seid=7745039378059831767)
11. [vscode检测到#include错误，请更新includePath。解决方法](https://blog.csdn.net/weixin_44718794/article/details/106751513)

2021年1月3日星期日 17:23