

调试 Linux 最早期的代码

Original 闪客 低并发编程 2022-05-09 17:30 Posted on 北京

收录于合集

#操作系统源码

43个

Linux 0.11 是 Linux 最早期的代码，非常适合作为第一款深入探索操作系统原理的代码。

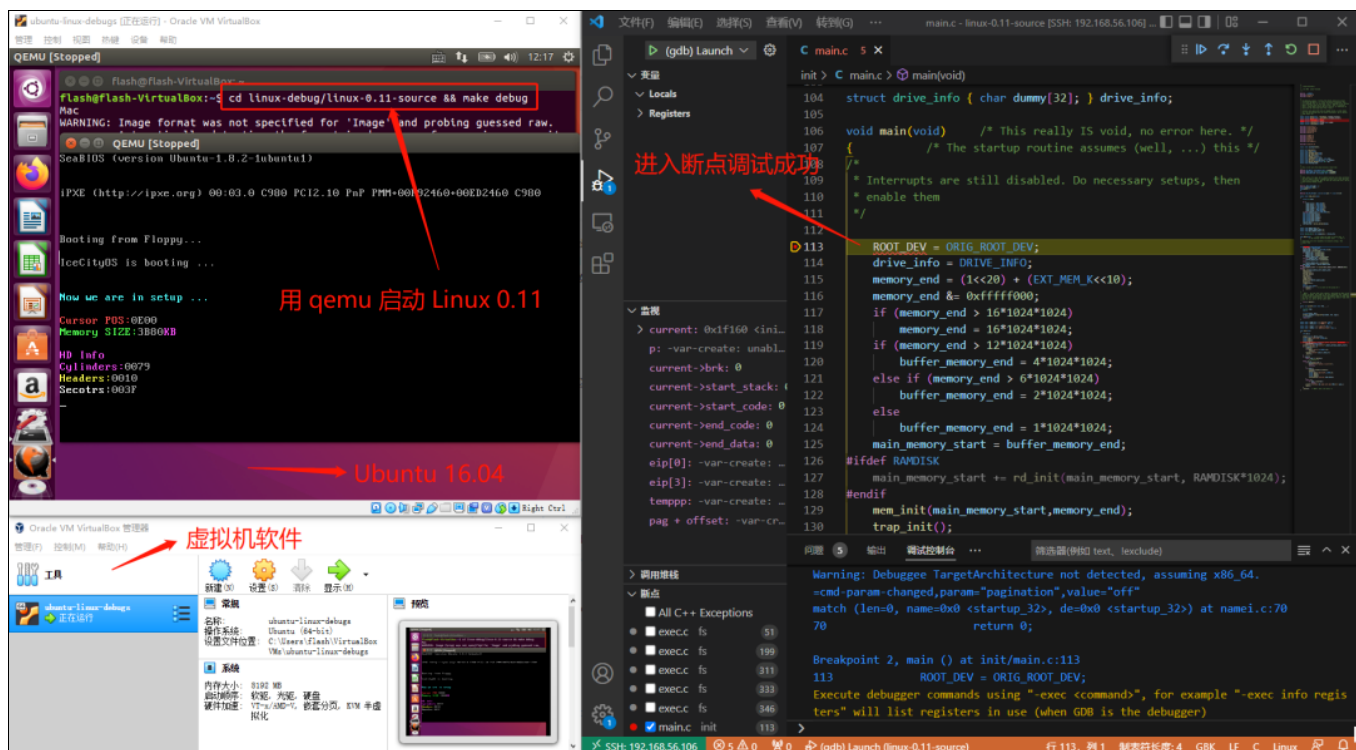
但同时，Linux 0.11 因为很多古老工具链的缺失，以及一些过时的文件格式比如 a.out，导致成功编译并运行它十分困难，更别说进行源码级别的 debug 调试了。

要想成功调试 Linux 0.11，需要进行很多改造，并依赖一些古老的工具链，对于仅仅是将 Linux 0.11 作为研究操作系统的手段的我们，无需花费精力自己去改造它，踩各种坑。

所以今天我就分享一下我调试 Linux 0.11 的一种方式，同时也为了我自己以后换电脑的时候方便照着自己的教程直接无脑搭建环境，那我们开始吧~

整体思路 and 效果

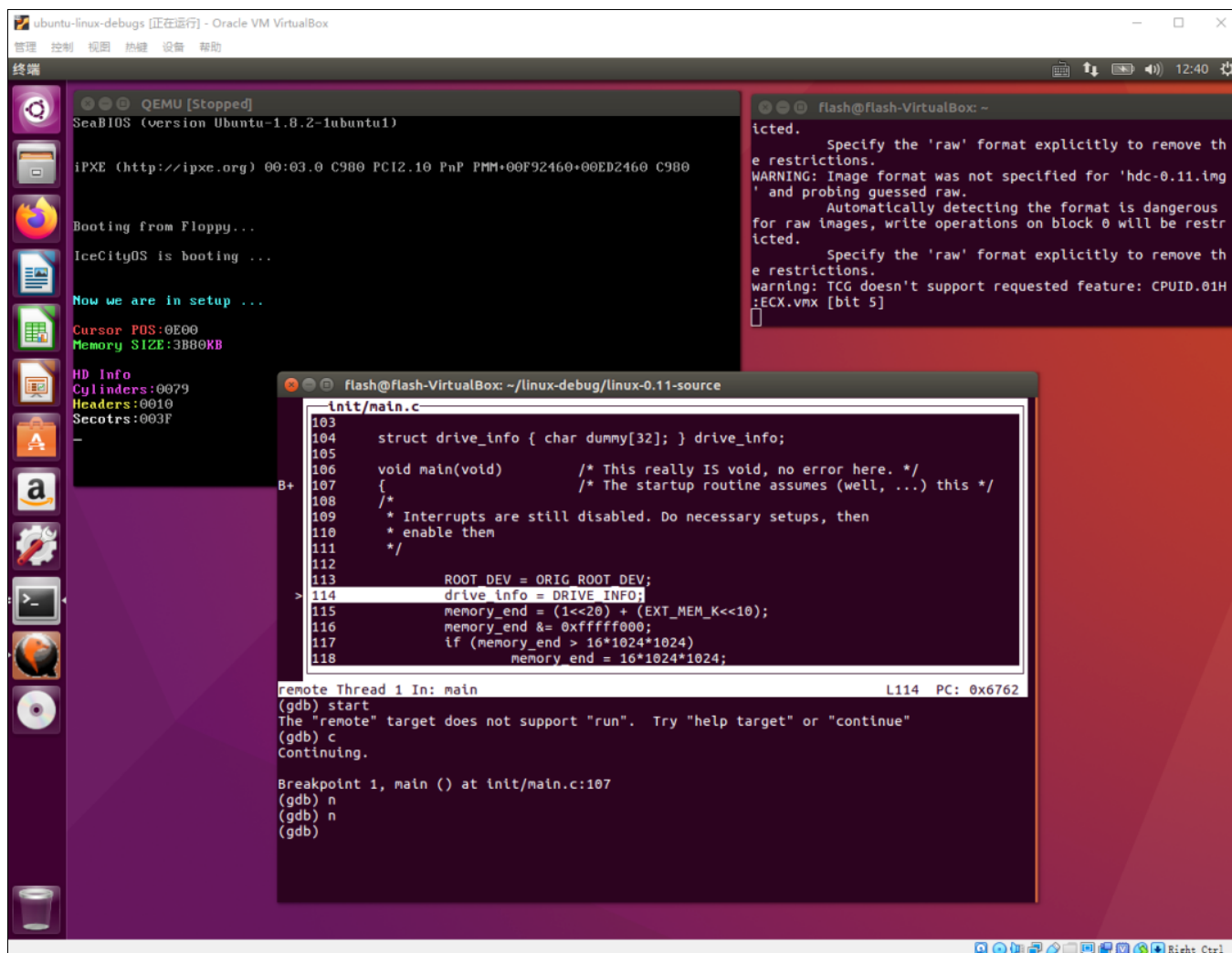
我用的方式是，在 windows 上，搞一个 Ubuntu 16.04 的虚拟机，在里面用 qemu 启动一个开启了调试的 Linux 0.11 系统，然后用本机的 vscode remote ssh 连接到虚拟机，并开启 gdb 调试，最终的效果如下。



这是最舒服的方式，因为 vscode 是本机的，完全不受虚拟机的影响，这也是我调试其他代码时比较喜欢的方式。

如果你有自己的豪华服务器，虚拟机也可以换成服务器，这样不但编译速度快，不消耗自己电脑的性能，同时也可以不受终端的影响，在家在公司都可以随时调试（方便摸鱼~）

当然如果你不需要这么直观，vscode 这一步也可以换成 gdb 命令行，在虚拟机里直接执行 gdb 相关命令即可。



下面我们就一步步来实现这个效果。

第一步：配置虚拟机

我用的虚拟机软件是

Oracle VM VirtualBox Version 6.0.8 Edition

官网是：

<https://ubuntu.com/>

下载页面是：

https://www.virtualbox.org/wiki/Download_Old_Builds_6_0

我这个版本的直接下载地址是：

<https://download.virtualbox.org/virtualbox/6.0.8/VirtualBox-6.0.8-130520-Win.exe>

安装的操作系统镜像是
ubuntu-16.04.7-desktop-amd64

官网是：

<https://ubuntu.com/>

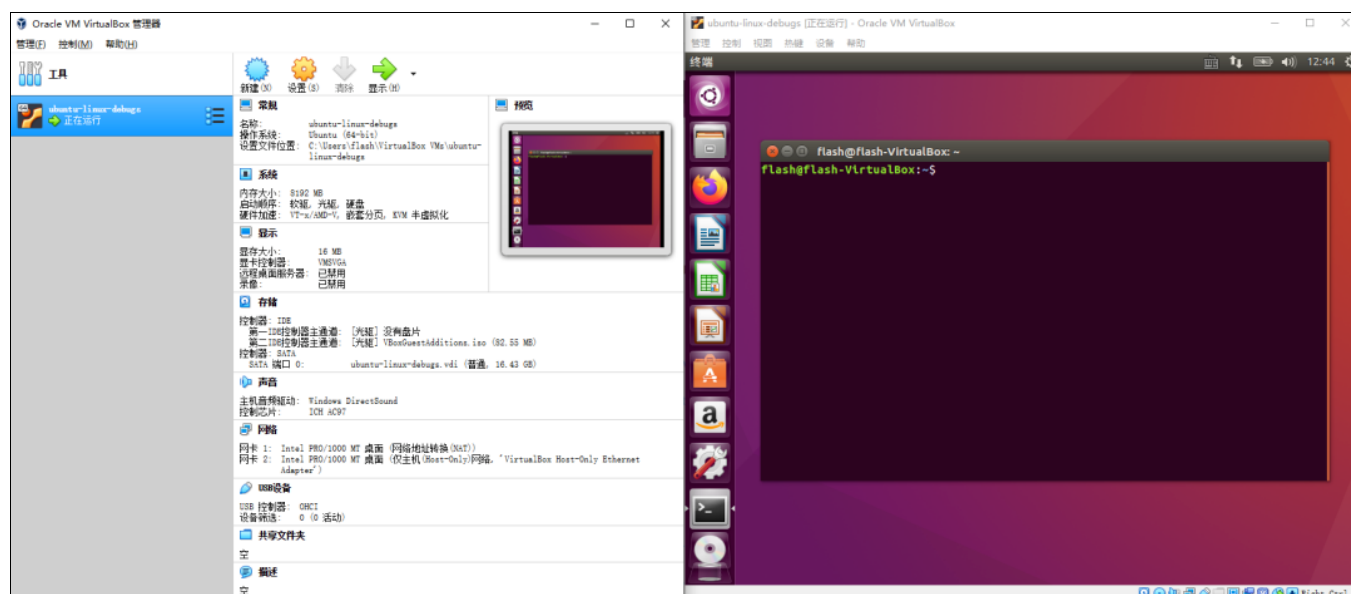
下载页面是：

<https://releases.ubuntu.com/xenial/>

我这个版本的直接下载地址是：

<https://releases.ubuntu.com/xenial/ubuntu-16.04.7-desktop-amd64.iso>

这个就不详细展开讲解了，最终达到这个效果就行。



第二步：安装 qemu

qemu 是模拟器，简单理解和虚拟机一样，用来当做真机启动 Linux 0.11 的。

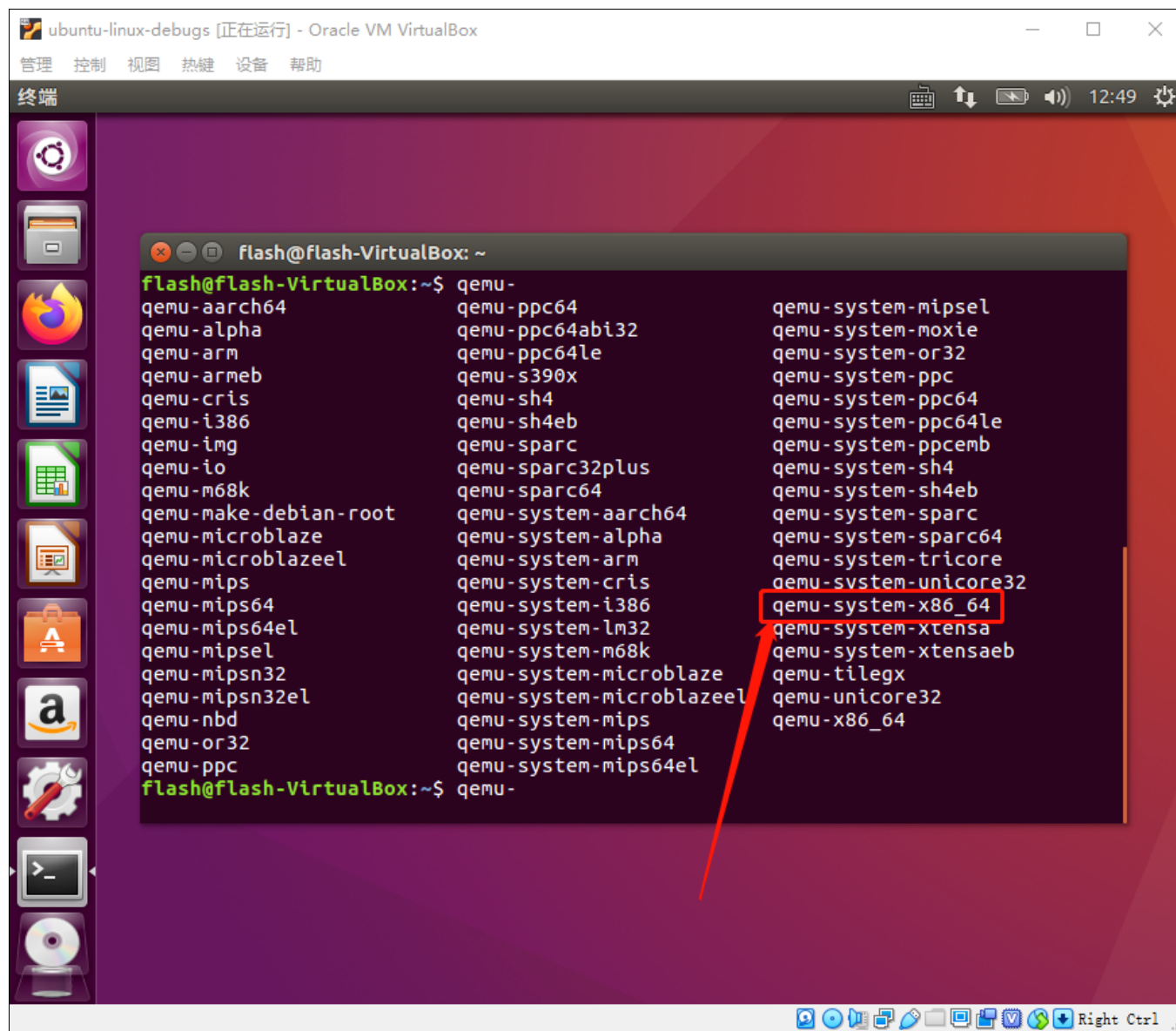
官网是这个：

<https://www.qemu.org/>

我们在刚刚的 Ubuntu 虚拟机里直接按照官方教程下载 qemu

sudo apt-get install qemu

下载好后，我们输入 **qemu-**，按两下 tab，查看下支持的体系结构。



看到有 **qemu-system-x86_64** 即可，一会我们会用这个来模拟启动 Linux 0.11。

第三步：下载并运行可调试的 Linux 0.11 源码

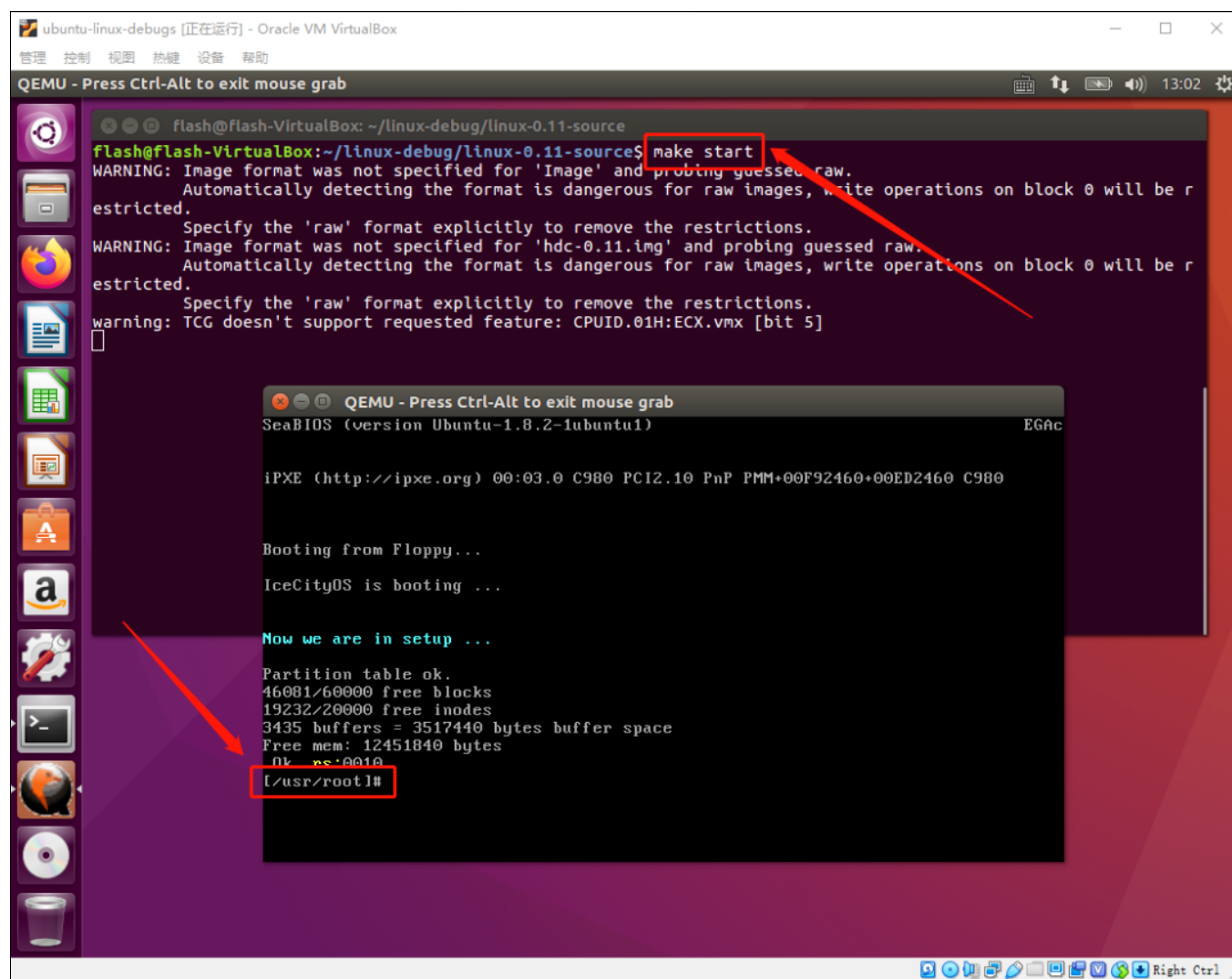
这一步直接下官网上的是不行的，因为那个依赖好多古老的工具链。

这里一般网上都是参考了赵炯老师为我们修改好的 Linux 0.11 源码，用现代的工具链即可构建，造福了广大热爱内核的开发者，我们直接拿来主义即可。

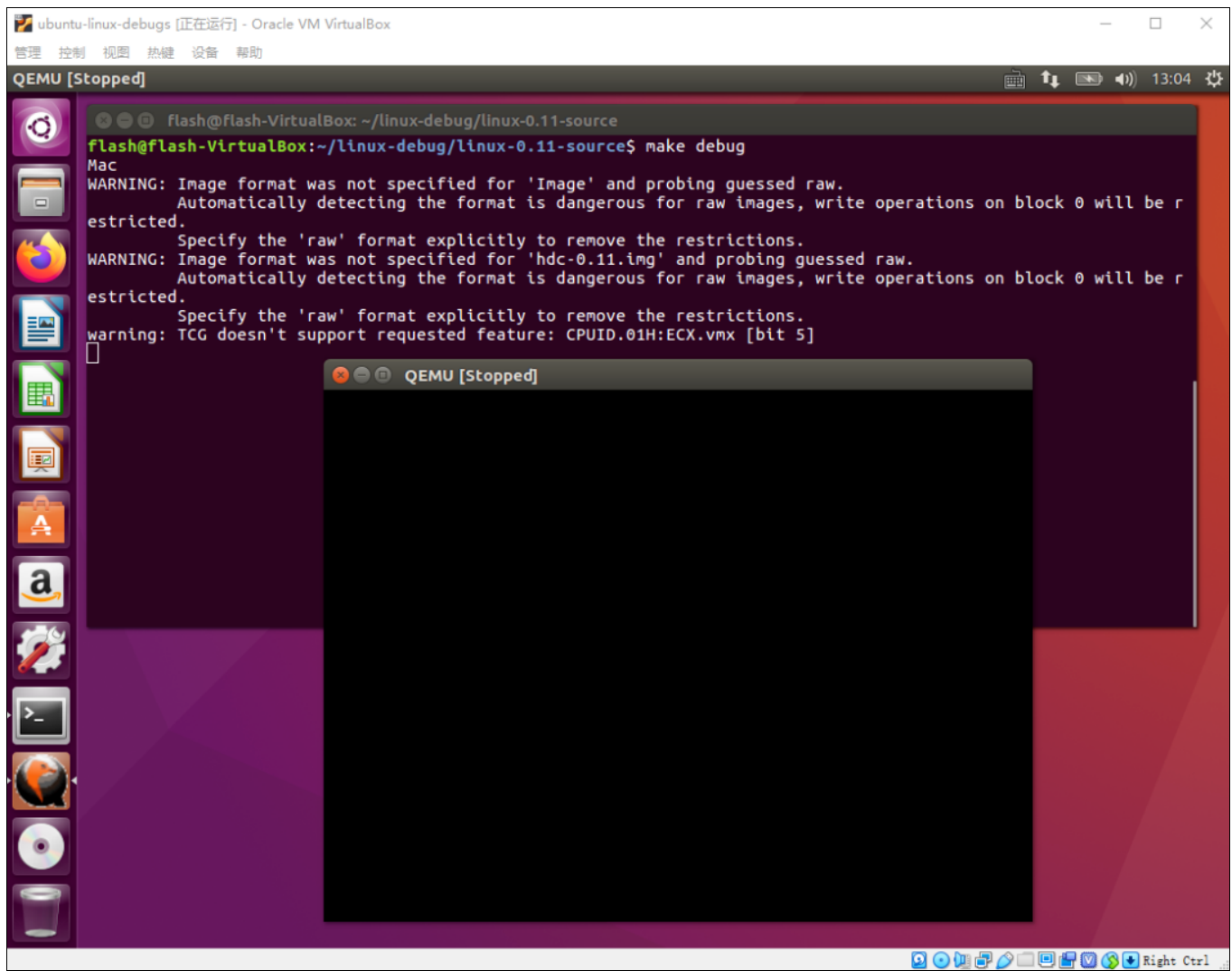
在赵炯老师准备好的源码的基础上，很多人又进行了二次改造，使得其可以一键 `qemu` 或 `bochs` 启动，这里我选择了仓库：

<https://github.com/yuan-xy/Linux-0.11>

直接把源码下载下来，进入根目录，输入命令 `make start` 就可以把 Linux 0.11 跑起来了。



如果想 debug，那么就以 debug 形式启动，输入命令 `make debug`，它会卡住不动。

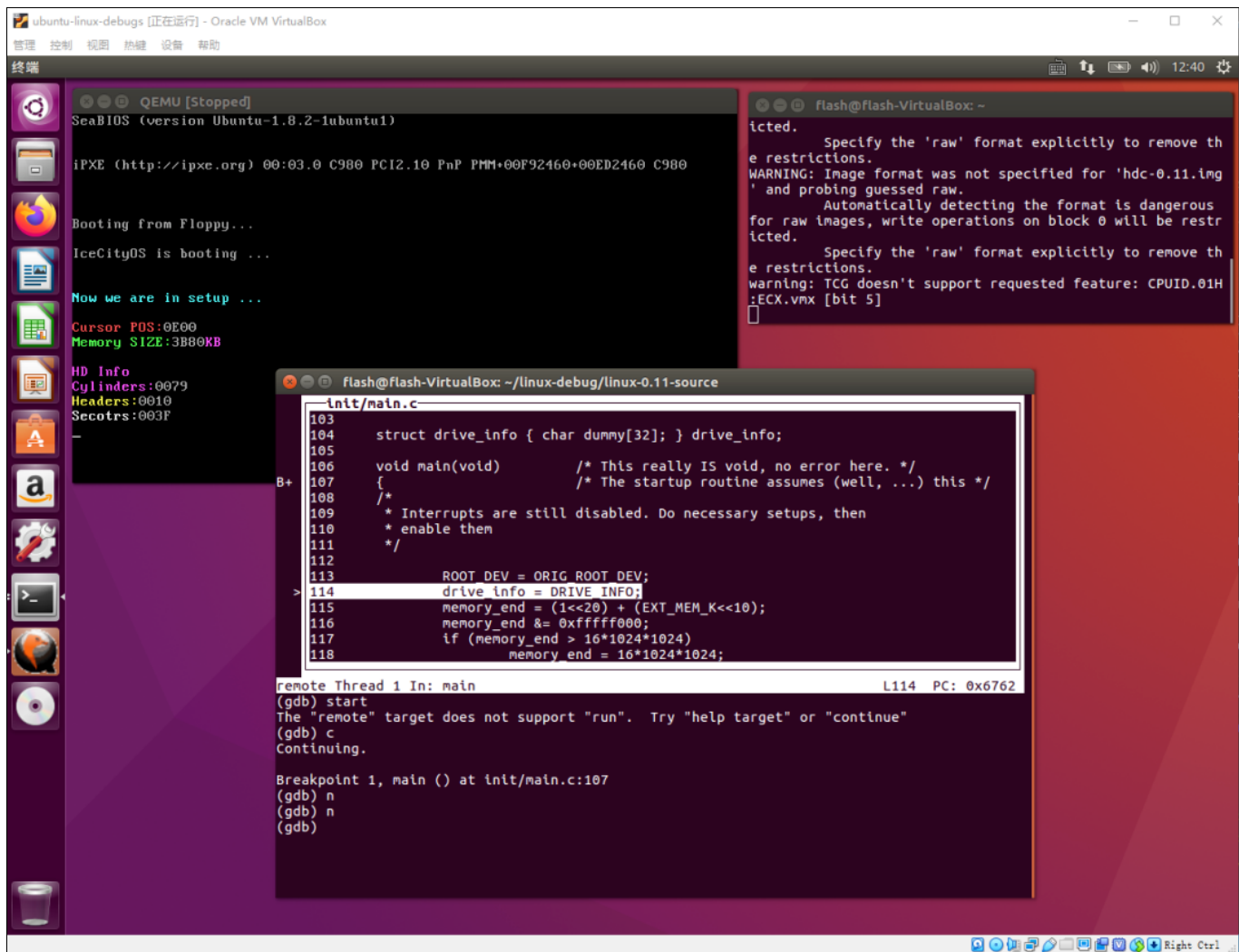


此时其实你就可以通过 gdb 进行调试了。

再开一个窗口，输入命令 **`gdb tools/system`**

然后 **`target remote :1234`**

就可以愉快地进行 gdb 调试了

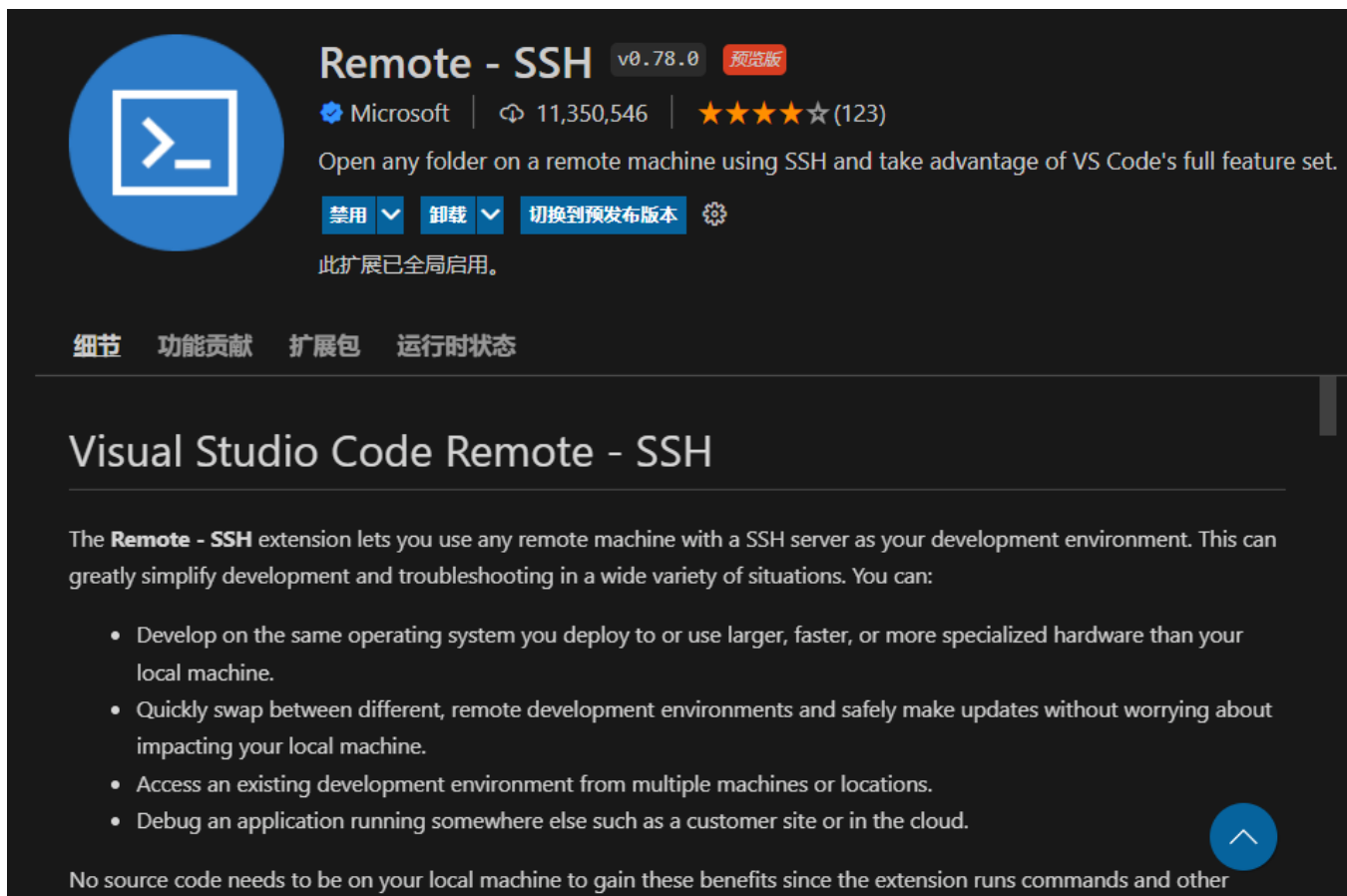


具体 gdb 怎么玩，就不展开讲解了。

第四步：通过 vscode 远程调试

当然，你也可以在虚拟机里用 vscode 进行本地调试，但我觉得不爽。


所以，在本机的 windows 里安装好 vscode，下载 **remote-ssh** 插件。



Remote - SSH v0.78.0 预览版

Microsoft | 11,350,546 | ★★★★★ (123)

Open any folder on a remote machine using SSH and take advantage of VS Code's full feature set.

[禁用](#) [卸载](#) [切换到预发布版本](#) 

此扩展已全局启用。

[细节](#) [功能贡献](#) [扩展包](#) [运行时状态](#)

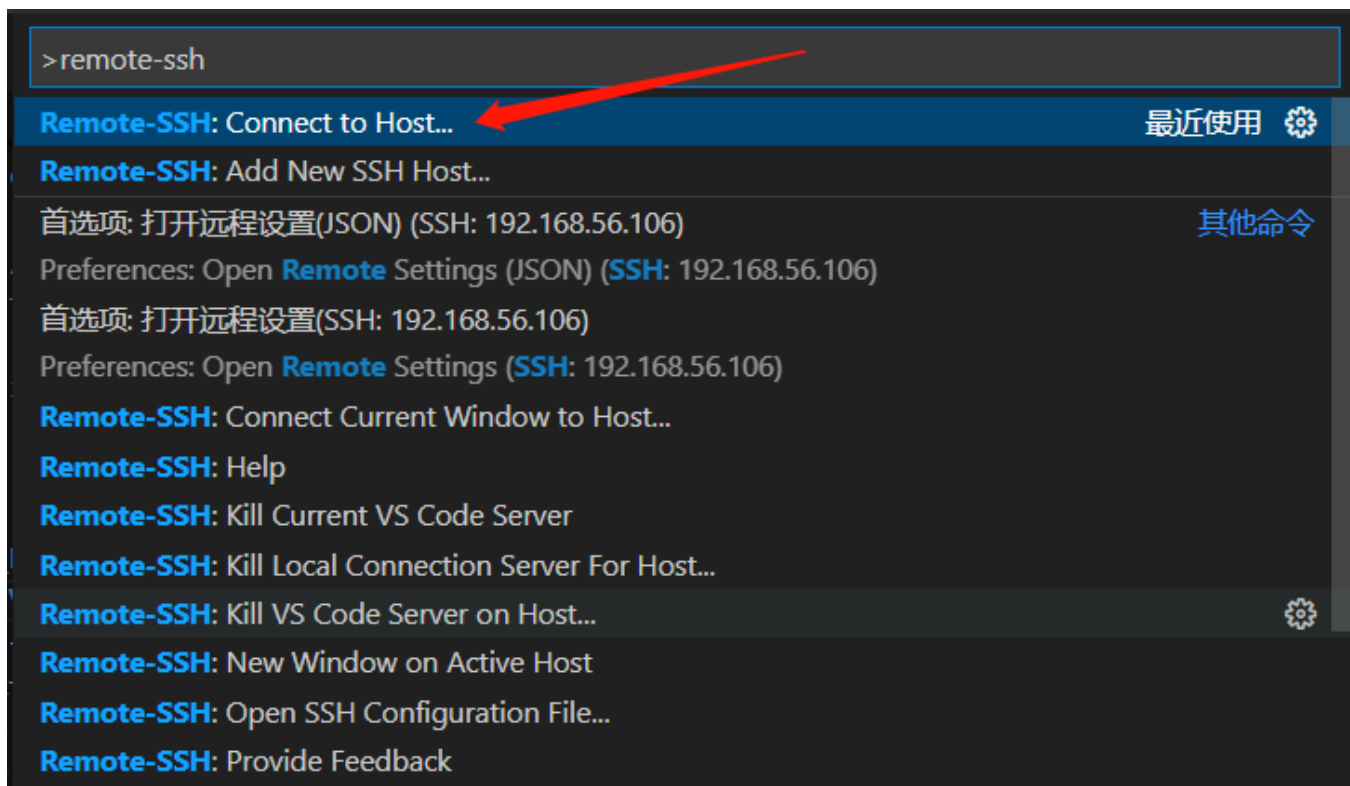
Visual Studio Code Remote - SSH

The **Remote - SSH** extension lets you use any remote machine with a SSH server as your development environment. This can greatly simplify development and troubleshooting in a wide variety of situations. You can:


- Develop on the same operating system you deploy to or use larger, faster, or more specialized hardware than your local machine.
- Quickly swap between different, remote development environments and safely make updates without worrying about impacting your local machine.
- Access an existing development environment from multiple machines or locations.
- Debug an application running somewhere else such as a customer site or in the cloud.

No source code needs to be on your local machine to gain these benefits since the extension runs commands and other

下载好后按下 **ctrl + p**, 输入 **>remote-ssh**, 找到 **Connect to Host**



>remote-ssh

Remote-SSH: Connect to Host... 最近使用 

Remote-SSH: Add New SSH Host...

首选项: 打开远程设置(JSON) (SSH: 192.168.56.106) 其他命令

Preferences: Open **Remote** Settings (JSON) (**SSH**: 192.168.56.106)

首选项: 打开远程设置(SSH: 192.168.56.106)


Preferences: Open **Remote** Settings (**SSH**: 192.168.56.106)

Remote-SSH: Connect Current Window to Host...

Remote-SSH: Help

Remote-SSH: Kill Current VS Code Server

Remote-SSH: Kill Local Connection Server For Host...

Remote-SSH: Kill VS Code Server on Host... 

Remote-SSH: New Window on Active Host

Remote-SSH: Open SSH Configuration File...

Remote-SSH: Provide Feedback

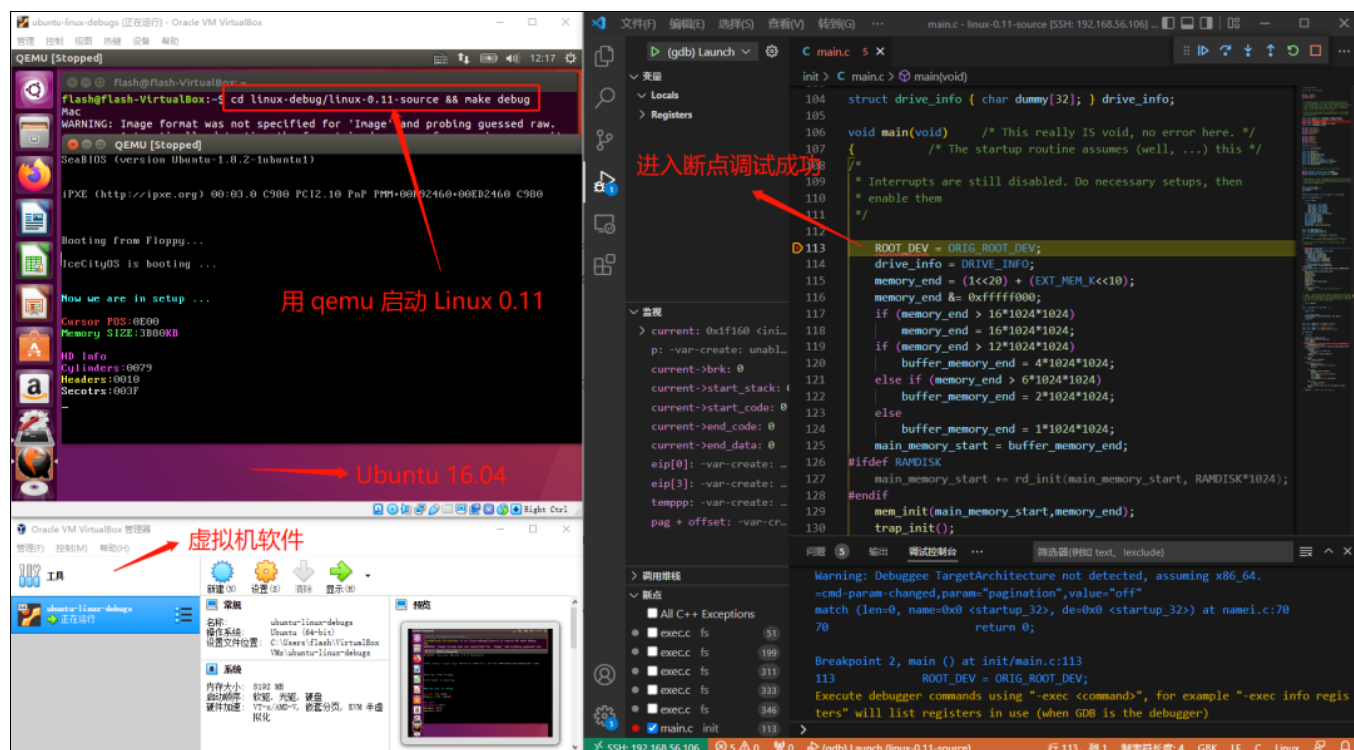
按照它提示的格式输入你的虚拟机 IP 和用户名，随后输入密码，即可远程连接到虚拟机。

之后点击菜单栏 **运行-启用调试**，在弹出的 **launch.json** 中做如下配置

```
.vscode > {} launch.json > Launch Targets > {} (gdb) Launch
1  {
2      "version": "0.2.0",
3      "configurations": [
4          {
5              "name": "(gdb) Launch",
6              "type": "cppdbg",
7              "request": "launch",
8              "program": "/home/flash/linux-debug/linux-0.11-source/tools/system",
9              "miDebuggerServerAddress": "127.0.0.1:1234",
10             "args": [],
11             "stopAtEntry": false,
12             "cwd": "${workspaceFolder}",
13             "environment": [],
14             "externalConsole": false,
15             "MIMode": "gdb"
16         }
17     ]
18 }
19 }
```

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(gdb) Launch",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/tools/system",
      "miDebuggerServerAddress": "127.0.0.1:1234",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb"
    }
  ]
}
```

配置好后保存，在 main 函数里打个断点，再次点击菜单栏 **运行-启用调试**，可以发现调试成功。



当然，记得每次 vscode 调试前，在虚拟机里先把 Linux 0.11 跑起来，就是执行命令 **make debug**。

这一步也可以配置到 vscode 里，但这一步没多少工作量，而且也不方便直观看到虚拟机里的行为，我就懒得弄了。

好了，这个教程到这里就结束了，这就是我调试 Linux 0.11 的其中一个办法，当然每个人可能都有自己喜欢的方式，只要适合自己就好。



低并发编程

战略上藐视技术，战术上重视技术

175篇原创内容

Official Account

收录于合集 #操作系统源码 43

上一篇

第35回 | 扒开 execve 的皮

下一篇

第36回 | 缺页中断

Read more

People who liked this content also liked

如何阅读 Go 源码

Golang技术分享



selenium操作方法大全

小K测试开发成长记



21张让你Python代码能力突飞猛进的速查表

Python丹卿

