

从零开始使用Vscode调试XV6



Atled

我唯一知道的就是我一无所知

103 人赞同了该文章

2022年 04月 20日 星期三 14:49:49 CST

准备Linux环境

不再赘述

Clone xv6 repo

下载xv6源代码

[github.com/mit-pdos/xv6...](https://github.com/mit-pdos/xv6-riscv)

```
git clone https://github.com/mit-pdos/xv6-riscv.git
```

```
cd xv6-riscv
```

Install Toolchains

需要的Tools如下:

- git
- make
- gcc
- perl
- qemu-system-riscv64
- riscv64-unknown-elf-gcc 或者 riscv64-linux-gnu-gcc
- riscv64-unknown-elf- 或者 riscv64-linux-gnu-
 - ld
 - objcopy
 - objdump
 - ... (这套东西一般叫binutils-riscv64-.....)
- gdb-multiarch

以ubuntu为例

```
sudo apt install binutils-riscv64-linux-gnu
sudo apt install gcc-riscv64-linux-gnu
sudo apt install gdb-multiarch
sudo apt install qemu-system-misc opensbi u-boot-qemu qemu-utils
```

实际上,你只需要跟着 make qemu 的报错,需要什么装什么就行了

make qemu

此时,在 xv6-riscv 目录下,执行 make qemu ,正常的话就进入了xv6.

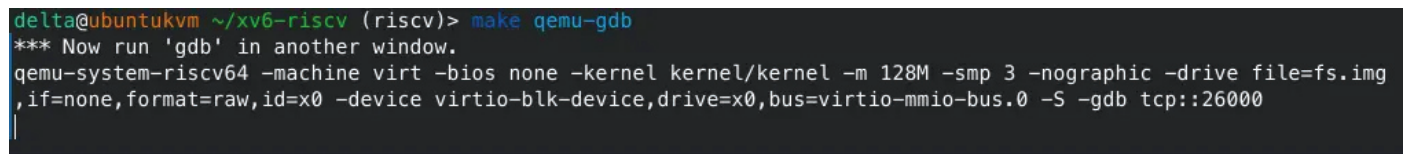
退出qemu:

1. first press Ctrl + A (A is just key a, not the alt key),
2. then release the keys,
3. afterwards press X.

完成这一步,才能继续

make qemu-gdb

此时,在 `xv6-riscv` 目录下,执行 `make qemu-gdb` ,进程会阻塞.

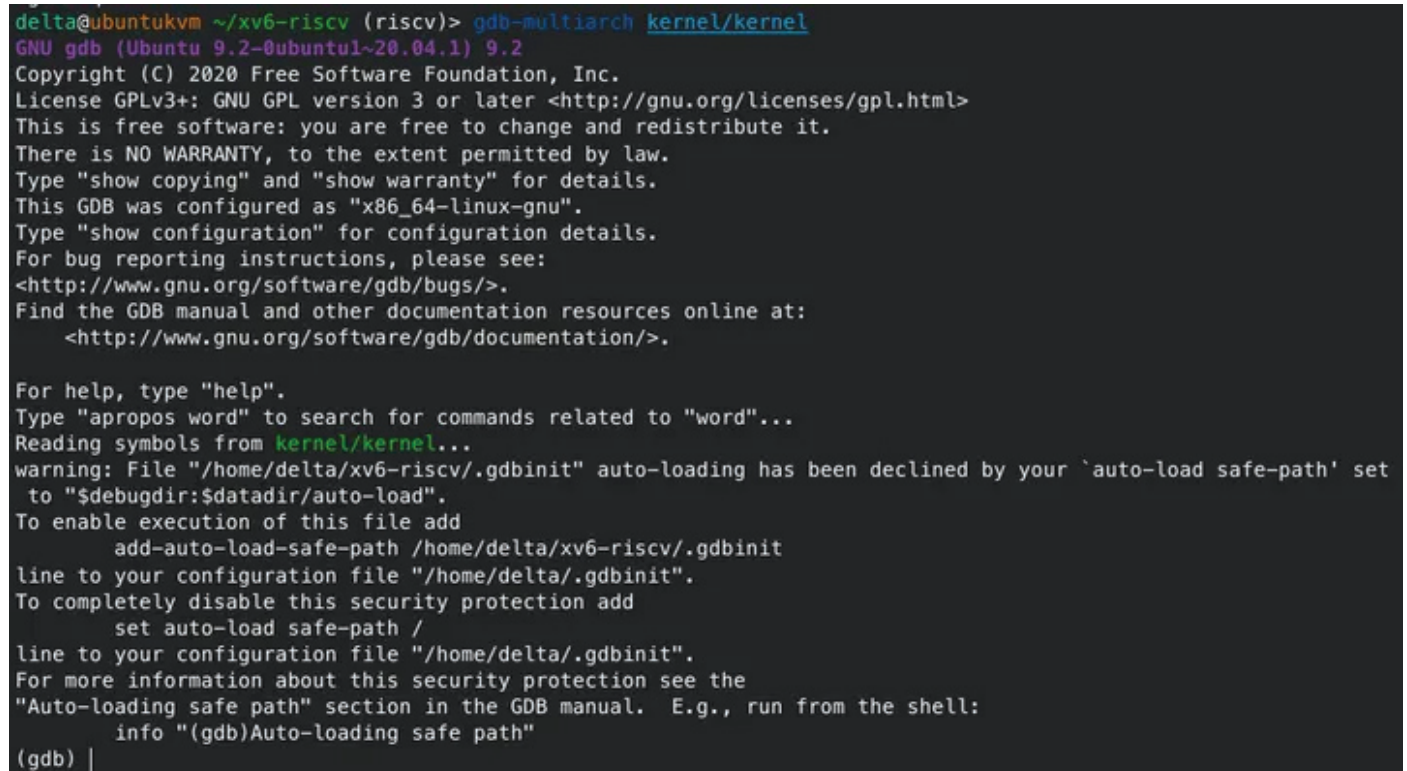


```
delta@ubuntukvm ~/xv6-riscv (riscv)> make qemu-gdb
*** Now run 'gdb' in another window.
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0 -S -gdb tcp::26000
```

Image

另开一个终端,在 `xv6-riscv` 目录下,执行 `gdb-multiarch kernel/kernel`

可能的问题



```
delta@ubuntukvm ~/xv6-riscv (riscv)> gdb-multiarch kernel/kernel
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel/kernel...
warning: File "/home/delta/xv6-riscv/.gdbinit" auto-loading has been declined by your `auto-load safe-path' set
to "$debugdir:$datadir/auto-load".
To enable execution of this file add
    add-auto-load-safe-path /home/delta/xv6-riscv/.gdbinit
line to your configuration file "/home/delta/.gdbinit".
To completely disable this security protection add
    set auto-load safe-path /
line to your configuration file "/home/delta/.gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
    info "(gdb)Auto-loading safe path"
(gdb) |
```

Image

`make qemu-gdb` 在当前目录生成了 `.gdbinit` ,具体见 `Makefile` Line 166.

`gdb` 会默认首先执行当前目录下的 `.gdbinit`

上图中这一步被阻止了.

warning: File "/home/delta/xv6-riscv/.gdbinit" auto-loading has been declined by your `auto-load safe-path' set to "\$debugdi

同样,正如他所说将

```
add-auto-load-safe-path /home/delta/xv6-riscv/.gdbinit
```

这一行加到 /home/delta/.gdbinit 这个文件里即可.

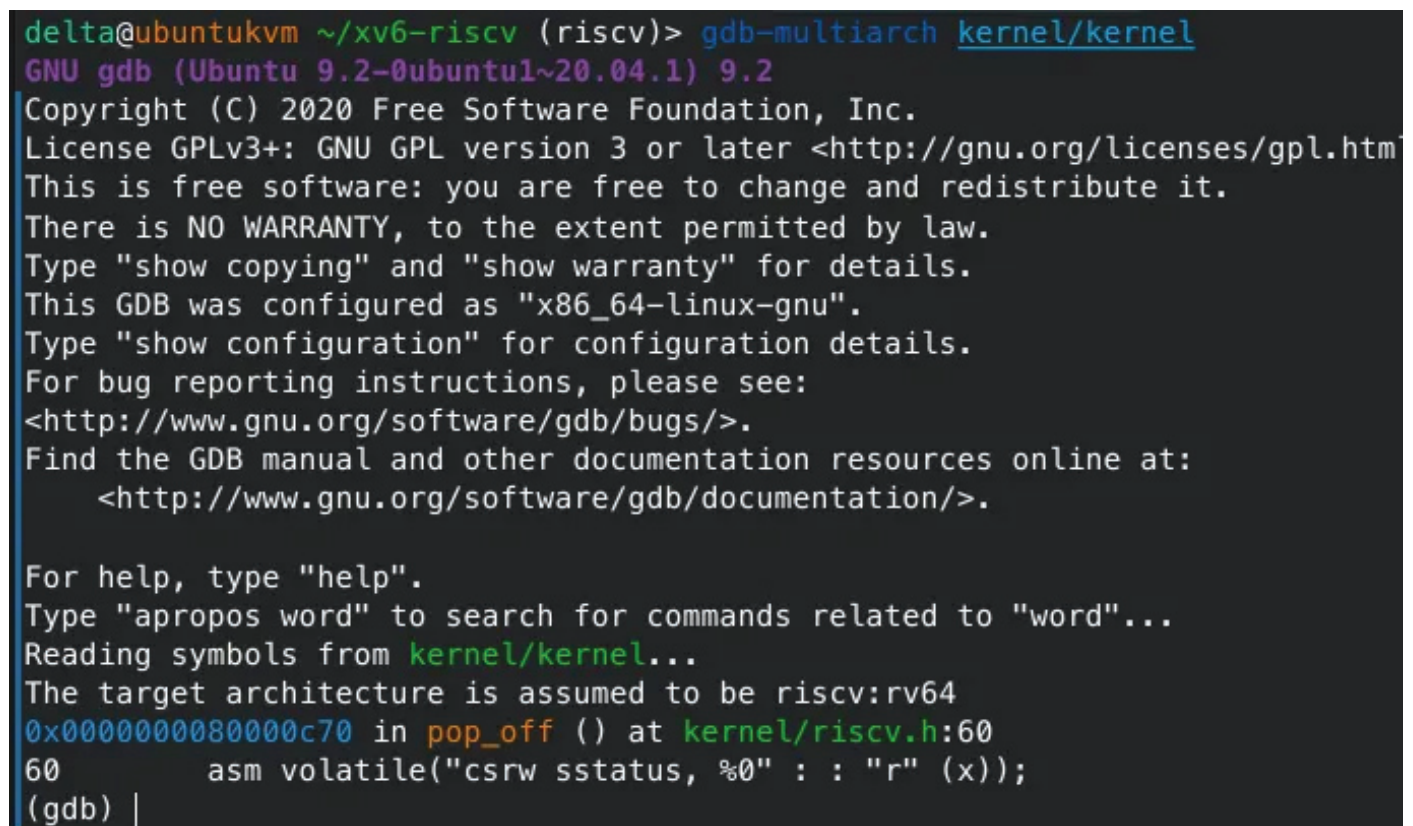
退出gdb快捷键 `ctrl+d` .

执行

```
echo "add-auto-load-safe-path /home/delta/xv6-riscv/.gdbinit" >> "/home/delta/.gdbinit"
```

或者vim手动编辑也行.

之后再次运行 `gdb-multiarch kernel/kernel`

A terminal window showing the execution of 'gdb-multiarch kernel/kernel'. The prompt is 'delta@ubuntukvm ~/xv6-riscv (riscv)'. The output shows the GNU gdb version 9.2, copyright information, and license details. It then shows the target architecture as riscv:rv64 and the current location in the code as '0x0000000080000c70 in pop_off () at kernel/riscv.h:60'. The command 'asm volatile("csrw sstatus, %0" : : "r" (x));' is shown, followed by the '(gdb) |' prompt.

```
delta@ubuntukvm ~/xv6-riscv (riscv)> gdb-multiarch kernel/kernel
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel/kernel...
The target architecture is assumed to be riscv:rv64
0x0000000080000c70 in pop_off () at kernel/riscv.h:60
60      asm volatile("csrw sstatus, %0" : : "r" (x));
(gdb) |
```

Image

一切正常

配置Vscode

安装拓展 C/C++

marketplace.visualstudio.com...

在 .vscode 目录下,创建以下两个文件

launch.json

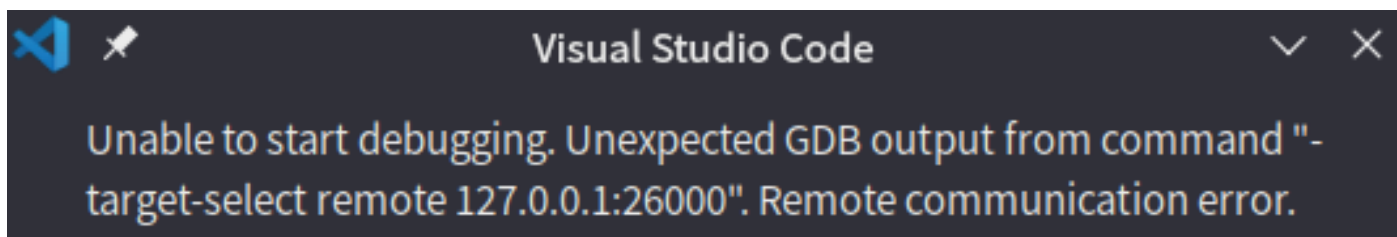
```
// xv6-riscv/.vscode/launch.json
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "xv6debug",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/kernel/kernel",
      "stopAtEntry": true,
      "cwd": "${workspaceFolder}",
      "miDebuggerServerAddress": "127.0.0.1:26000", //见.gdbinit 中 target remote xxxx:xx
      "miDebuggerPath": "/usr/bin/gdb-multiarch", // which gdb-multiarch
      "MIMode": "gdb",
      "preLaunchTask": "xv6build"
    }
  ]
}
```

tasks.json

```
// xv6-riscv/.vscode/tasks.json
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "xv6build",
      "type": "shell",
      "isBackground": true,
      "command": "make qemu-gdb",
      "problemMatcher": [
        {
          "pattern": [
            {
              "regexp": ".",
              "file": 1,
              "location": 2,
              "message": 3
            }
          ],
          "background": {
            "beginsPattern": ".*Now run 'gdb' in another window.",
            // 要对应编译成功后,一句echo的内容. 此处对应 Makefile Line:170
            "endsPattern": "."
          }
        }
      ]
    }
  ]
}
```

按F5即可开始调试.

可能会发现报错如下



Target disconnected.: Connection reset by peer.

取消

打开“launch.json”

或者

```
> Executing task: make qemu-gdb <
```

```
*** Now run 'gdb' in another window.
```

```
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0 -S -gdb tcp::26000
```

```
qemu-system-riscv64: QEMU: Terminated via GDBstub
```

终端将被任务重用，按任意键关闭。

[stackoverflow.com/quest...](https://stackoverflow.com/questions/44714447/gdb-terminates-qemu-system-riscv64)

GDB automagically loads ~/.gdbinit.

so when you load .gdbinit via --command=~/.gdbinit

it runs the script twice,

when it gets to the 2nd invocation of target remote localhost:1234

gdb hangs up its initial connection, qemu quits,

then gdb fails to reconnect to it because it is no longer running.

这是因为 .gdbinit 中有 target remote 127.0.0.1:26000 ,这个文件会被gdb最先执行一遍.

此外,我们在 launch.json 中指定了 "miDebuggerServerAddress": "127.0.0.1:26000" ,同一个remote address被配置了两次.

只需要把 .gdbinit 中的注释掉即可

```
set confirm off
set architecture riscv:rv64
@REM target remote 127.0.0.1:26000
symbol-file kernel/kernel
set disassemble-next-line auto
set riscv use-compressed-breakpoints yes
```

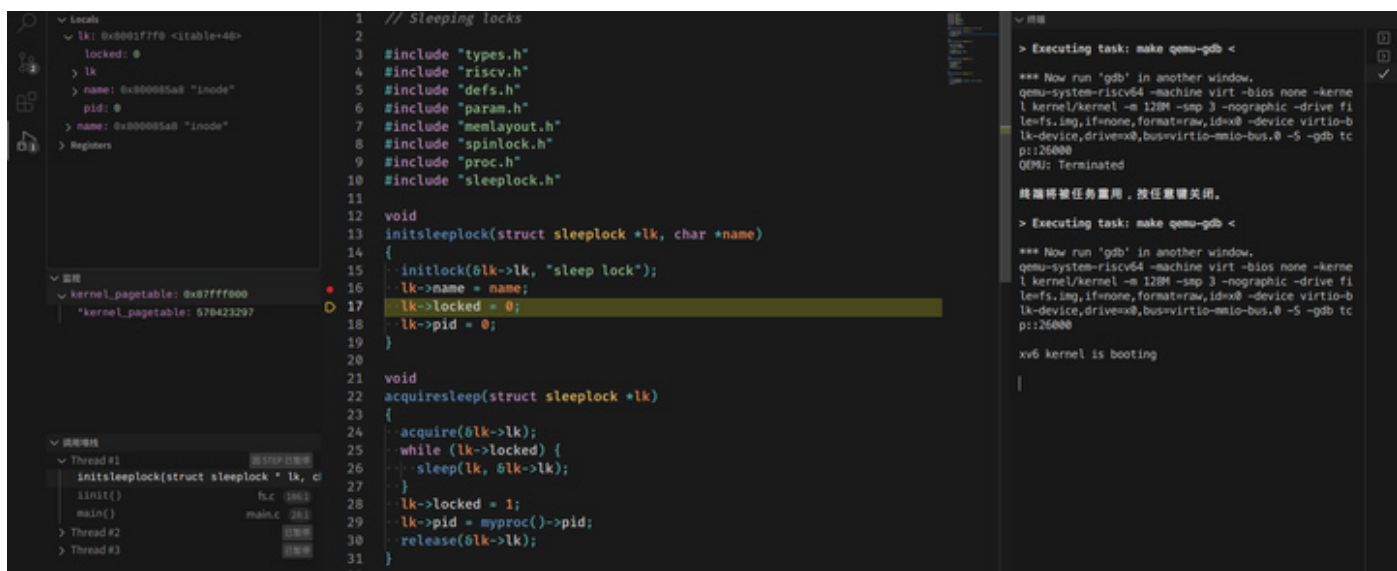
@REM 就是 .gdbinit 的注释符号.

要是你使用命令行gdb,记得再改回来.

最后

jyy yyds [bilibili.com/video/av63...](https://www.bilibili.com/video/av63...)





Image

```
./
├── .dir-locals.el
├── .editorconfig
├── .gdbinit.tmpl-riscv
├── .git
│   ├── .....
├── .gitignore
├── kernel
│   ├── bio.c
│   └── .....
├── LICENSE
├── Makefile
├── mkfs
│   └── mkfs.c
├── README
├── user
│   ├── cat.c
│   └── .....
└── .vscode
    ├── launch.json
    └── tasks.json
```

本文使用 [Zhihu On VSCode](#) 创作并发布

2022年 10月 09日 星期日 14:25:46 CST 更新

配置IntelliSense实现完美跳转

1. 安装 [bear](#)
2. 在xv6目录下执行 `make clean && bear -- make qemu` , 将会生成一个 `compile_commands.json`
3. 将 `compile_commands.json` 移动到 `.vscode` 目录下
4. 在 `.vscode` 目录下新建 `c_cpp_properties.json` 文件

```
// xv6-riscv/.vscode/c_cpp_properties.json
{
```



```
"configurations": [
  {
    "name": "Linux",
    // "includePath": [],
    // "defines": [],
    // "compilerPath": "/usr/bin/riscv64-linux-gnu-gcc",
    // "cStandard": "gnu17",
    // "intelliSenseMode": "${default}",
    "compileCommands": "${workspaceFolder}/.vscode/compile_commands.json"
    // "configurationProvider": "ms-vscode.makefile-tools"
  }
],
"version": 4
}
```

至此，应该可以实现F12自由跳转了。

编辑于 2022-10-09 14:44

操作系统 VSCode MIT 公开课程

▲ 赞同 103 ▼ ● 26 条评论 ↗ 分享 ♥ 喜欢 ★ 收藏 📄 申请转载 ...

写下你的评论...

26 条评论

默认 最新

OMO

请问要如何在用户程序中添加断点呢，根本不会在断点停下来

2022-09-21

● 回复 ♥ 1

量产型扎古X

我是改了这里"program": "\${workspaceFolder}/user/_find" 能够断点。_find是我写的find.c编译后输出的文件吧，Makefile里面UPROGS那里是啥就写啥

01-05

● 回复 ♥ 1

ZZZ

控制臺exec添加文件即可

01-24

● 回复 ♥ 喜欢

展开其他 1 条回复 >

龙的传人科龙

mac用户如何调试? 🤖

03-06

● 回复 ♥ 喜欢

transactionlayer

请教一下这个版本怎么grade呢?

2023-09-09

● 回复 ♥ 喜欢

潜伏

GitHub里的感觉不全，很多分支都没有。还是clone mit自己的git服务器吧

2023-09-16

● 回复 ♥ 喜欢

transactionlayer

换了个官方最新版本，有打分脚本

2023-09-09

● 回复 ♥ 喜欢

joy

我生成只有 [

{

"arguments": [

"/usr/bin/gcc",

```
"-c",
"-Werror",
"-Wall",
"-l.",
"-o",
"mkfs/mkfs",
"mkfs/mkfs.c"
],
"directory": "/Users/xx/dev/data/xv6-riscv",
"file": "/Users/xx/dev/data/xv6-riscv/mkfs/mkfs.c",
"output": "/Users/xx/dev/data/xv6-riscv/mkfs/mkfs"
}
]
```

2023-06-20

● 回复 ● 喜欢



胡子叔叔

牛啊哥

2023-04-26

● 回复 ● 喜欢



米小米upup

谢谢!

2023-03-22

● 回复 ● 喜欢



David

感谢老哥，真保姆级教程，it works!

2023-03-07

● 回复 ● 喜欢



格栅套头

牛逼 完整全面 好厉害

2023-02-28

● 回复 ● 喜欢



sunshine

大佬，你好，为啥我把这些json文件放在xv6的.vscode目录下，无法弹出调试窗口呢，也就是无法开始调试??? 求指教

2022-12-21

● 回复 ● 喜欢



森林

Ubuntu 20.04 执行make clean && bear -- make qemu 后，产生的文件是compile_commands.events.json，有1200多行。请问这个要怎么解决?

2022-11-22

● 回复 ● 喜欢



森林 ▸ Atled

感谢，我再试试

2022-11-24

● 回复 ● 喜欢



Atled 作者

compile_commands.events.json是个中间文件，退出qemu后，就生成了compile_commands.json

2022-11-23

● 回复 ● 喜欢

展开其他 1 条回复

点击查看全部评论

写下你的评论...