

vscode C++ 调试环境搭建

1. 开始（简单了解即可）

在项目中为们可以看到他有提供了一个debug版本的选项,如果我们要对其进行调试，就必须使用这个版本。

```
meson setup --buildtype debug build_debug
meson compile -C build_debug
```

生成debug 版本后就可以直接用GDB 进行调试了。

执行以下命令：

```
gdb ./search/build_debug/atsipp.exe
```

```
PS E:\Company_work\delay-replanning> gdb ./search/build_debug/atsipp.exe
GNU gdb (GDB) 15.2
Copyright (C) 2024 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-w64-mingw32".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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 ./search/build_debug/atsipp.exe...
(gdb)
```

进行以下界面就说明了你已经进入了GDB调试环境了。

执行以下命令：

```
run --edgegraph ../output --start t-405B --goal t-401A
```

```
Reading symbols from ./search/build_debug/atsipp.exe...
(gdb) run --edgegraph output --start t-405B --goal t-401A
Starting program: E:\Company_work\delay-replanning\search\build_debug\atsipp.exe
--edgegraph output --start t-405B --goal t-401A
[New Thread 2316.0x6578]
[New Thread 2316.0x64ec]
[New Thread 2316.0xba4]
nodes read
Error: unable to find safe starting state: tried to find t-405B at time t=0
[Thread 2316.0x64ec exited with code 4294967295]
[Thread 2316.0x6578 exited with code 4294967295]
[Thread 2316.0xba4 exited with code 4294967295]
[Inferior 1 (process 2316) exited with code 03777777777]
```

可以看到，程序执行了，提示没有找到列车。

现在我们可以给他添加断点：

```
break main.cpp:25
```

这个的意思是在main.cpp 文件下的第25行添加一个断点，然后在执行运行命令

```
(gdb) break main.cpp:25
Breakpoint 1 at 0x7ff76cc11723: file ../main.cpp, line 26.
(gdb) run --edgegraph output --start t-405B --goal t-401A
Starting program: E:\Company_work\delay-replanning\search\build_debug\atsipp.exe -
-edgegraph output --start t-405B --goal t-401A
[New Thread 26540.0x6434]
[New Thread 26540.0x10ac]
[New Thread 26540.0x3210]

Thread 1 hit Breakpoint 1, main (argc=7, argv=0xf13eb0) at ../main.cpp:26
26         po::variables_map vm;
(gdb)
```

可以看到代码在这里被暂停了。

```
26         po::variables_map vm;
(gdb) next
27         po::store(po::parse_command_line(argc, argv, desc), vm);
(gdb) print(vm)
$1 = <incomplete type>
(gdb) list
22         ("agentSpeed,a", po::value<double>()->default_value(15.0), "Travel
ing speed of the agent.")
23         ("walkingSpeed,w", po::value<double>()->default_value(1.0), "Walki
ng speed for reversing train.")
24         ("lookups,l", po::value<long>()->default_value(100), "Number of lo
okups to test repeat")
25         ;
26         po::variables_map vm;
27         po::store(po::parse_command_line(argc, argv, desc), vm);
28         po::notify(vm);
29
30         if (vm.count("help")){
31             std::cout << desc << std::endl;
(gdb)
```

- next: 下一句
- print(变量名): 打印变量
- list: 查看代码块
- ...

剩下的一些命令可以自行网络上了解，这就是使用原始GDB进行调试的方法。

使用这种方式可以看到，不直观也很不方便，所幸vscode 提供了更加便捷的调试方式（本质上用的还是GDB,只不过提供了更加便捷的方式）

2. 配置VScode

1.首先我们要做的就是源代码中打断点，打断点的方式如图所示：

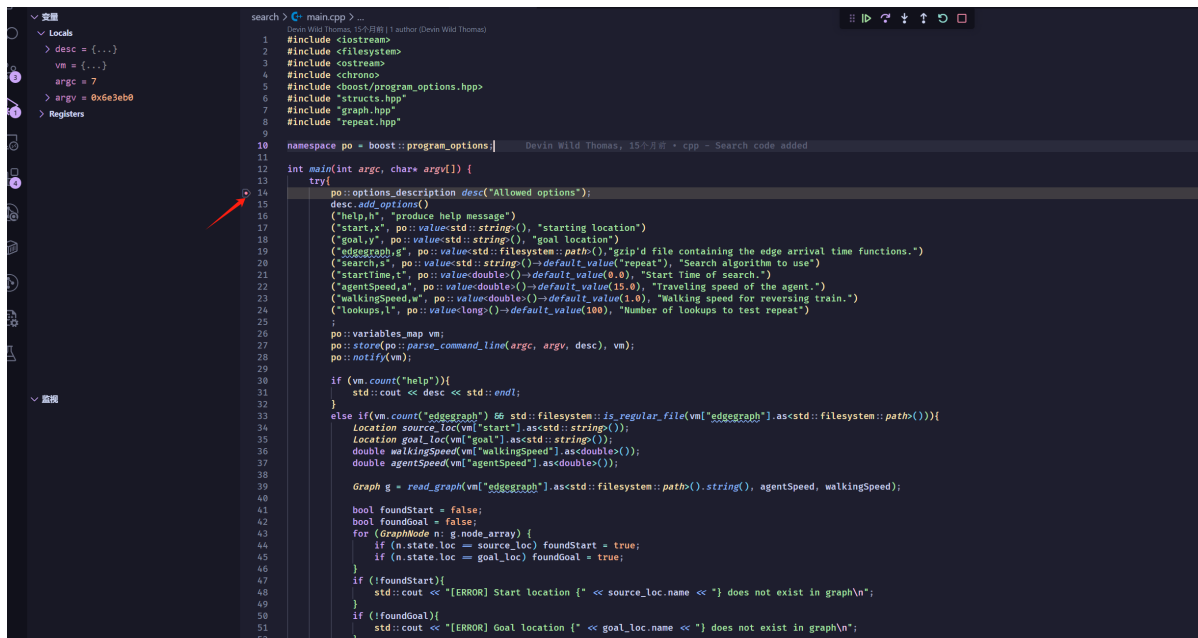
```
12 int main(int argc, char* argv[]) {
13     try{
14         po::options_description desc("Allowed options");
15         desc.add_options()
16             ("help,h", "produce help message")
17             ("start,x", po::value<std::string>(), "starting location")
18             ("goal,y", po::value<std::string>(), "goal location")
19             ("edgegraph,g", po::value<std::filesystem::path>(), "gzip'd file containing the edge arrival time functions.")
20             ("search,s", po::value<std::string>()>default_value("repeat"), "Search algorithm to use")
21             ("starttime,t", po::value<double>()>default_value(0.0), "Start Time of search.")
22             ("agentSpeed,a", po::value<double>()>default_value(15.0), "Traveling speed of the agent.")
23             ("walkingSpeed,w", po::value<double>()>default_value(1.0), "Walking speed for reversing train.")
24             ("lookups,l", po::value<long>()>default_value(100), "Number of lookups to test repeat")
25         ;
26         po::variables_map vm;
27         po::store(po::parse_command_line(argc, argv, desc), vm);
28         po::notify(vm);
29         Devin Wild Thomas, 15个月前 · cpp - Search code added
30         if (vm.count("help")){
31             std::cout << desc << std::endl;
32         }
33     }
```

2. 然后我们需要配置 launch.json 文件，在项目根目录创建 .vscode/launch.json 文件。

```
{
  "version": "0.2.0",
  // 重点是这部分
  "configurations": [
    // 这里每一个对象对应的就是vscode的一个调试任务
    {
      "name": "调试 atsipp (Meson Debug)",
      "type": "cppdbg",
      "request": "launch",
      // ${workspaceFolder} 表示工作区，以.vscode 上级文件夹为根目录下都是工作区
      // 这个关键字表示这就是我们要调试的程序
      "program": "${workspaceFolder}/search/build_debug/atsipp.exe",
      // 这个就是我们程序后面携带的参数
      "args": [
        "--edgegraph", "../output",
        "--start", "t-405B",
        "--goal", "t-401A"
      ],
      "stopAtEntry": false,
      // 这个要重点理解一下，他就是相当于我们这个命令执行的目录，就像我们前面的cd
      // ../search
      // 然后执行命令，所对应的output 文件生成在根目录下所以就得上面的arg部分就得变成../output
      "cwd": "${workspaceFolder}/search",

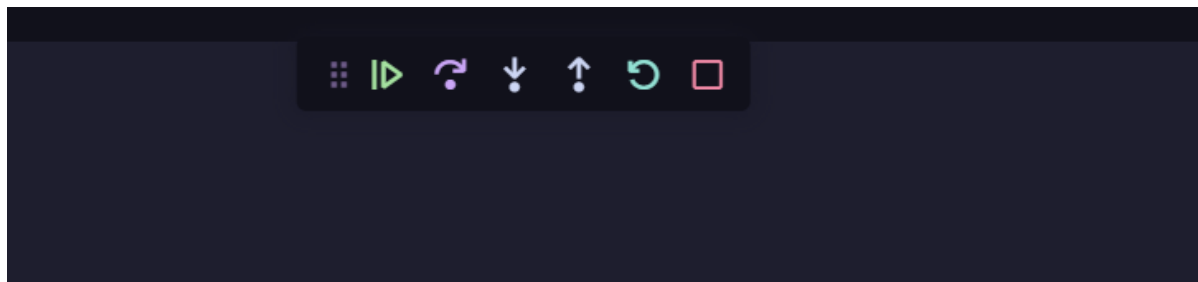
      // 剩下这些为默认配置，如果向详细了解可以去网上查看教程
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "miDebuggerPath": "gdb.exe",
      "setupCommands": [
        {
          "description": "为 gdb 启用漂亮打印",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ]
    }
  ]
}
```

配置完成以上部分之后，我们就可以在文件下直接进行调试了。在cpp文件下输入 F5



可以看到程序就在这里被停住了。

在这个上面我们可以看到六个标志



他们分别是：

- 执行到下一个断点
- 下一步
- 进入到方法
- 跳出方法
- 重新执行
- 退出

现在就可以开始调试了！😄😄😄😄

但是现在又有一个问题，如果我改动了文件，我就得重新执行一遍最先开始的编译命令，每次这样就很麻烦，有没有什么更加方便的方法能够快速的调试呢。

```
meson setup --buildtype debug build_debug
meson compile -C build_debug
```

3. 配置task任务

在 .vscode 文件夹下面创建 task.json 文件：

```
{
```

```

"version": "2.0.0",
"tasks": [
  {
    "label": "Meson Setup (Debug)",
    "type": "shell",
    "command": "meson setup --buildtype debug build_debug",
    "options": {
      "cwd": "${workspaceFolder}/search"
    },
    "problemMatcher": [],
    "group": {
      "kind": "build",
      "isDefault": true
    }
  },
  {
    "label": "Meson Compile (Debug)",
    "type": "shell",
    "command": "meson compile -C build_debug",
    "problemMatcher": [],
    "dependsOn": ["Meson Setup (Debug)"],
    "options": {
      "cwd": "${workspaceFolder}/search"
    },
    "group": {
      "kind": "build",
      "isDefault": false
    }
  }
]
}

```

在 `launch.json` 文件下新增字段：

`"preLaunchTask": "Meson Compile (Debug)"` // 这个其实就是告诉执行调试前先执行编译任务，这个名称对应的就是上面的 `task` 里面的 `label`

```

"version": "0.2.0",
"configurations": [
  {
    "name": "调试 atsipp (Meson Debug)",
    "type": "cppdbg",
    "request": "launch",
    "program": "${workspaceFolder}/search/build_debug/atsipp.exe",
    "args": [
      "--edgegraph",
      "../output",
      "--start",
      "t-405B",
      "--goal",
      "t-401A"
    ],
    "stopAtEntry": false,
    "cwd": "${workspaceFolder}/search",
    "environment": [],
    "externalConsole": false,
    "MIMode": "gdb",
    "miDebuggerPath": "gdb.exe", // 如果用的是 MinGW, 通常为 mingw32-gdb.exe
    "setupCommands": [
      {
        "description": "为 gdb 启用漂亮打印",
        "text": "-enable-pretty-printing",
        "ignoreFailures": true
      }
    ],
    "preLaunchTask": "Meson Compile (Debug)"
  }
]

```

然后就可以修改代码，然后 F5，就可以看到测试成功了。

```
37     } else if (vm.count("edgegraph") &&
38                std::filesystem::is_regular_file(
39                    vm["edgegraph"].as<std::filesystem::path>())) {
40         Location source_loc(vm["start"].as<std::string>());
41         Location goal_loc(vm["goal"].as<std::string>());
42         double walkingSpeed(vm["walkingSpeed"].as<double>());
43         double agentSpeed(vm["agentSpeed"].as<double>());
44
45         std::cout << "[DEBUG] 测试测试" << std::endl;
46
47         Graph g = read_graph(vm["edgegraph"].as<std::filesystem::path>().string(),
48                               agentSpeed, walkingSpeed);
49
50         Devin Wild Thomas, 15个月前 • cpp - Search code added
51         bool foundStart = false;
52         bool foundGoal = false;
53         for (GraphNode n : g.node_array) {
54             if (n.state.loc == source_loc)
55                 foundStart = true;
56             if (n.state.loc == goal_loc)
57                 foundGoal = true;
58         }
59         if (!foundStart) {
60             std::cout << "[ERROR] Start location {" << source_loc.name
61                 << "} does not exist in graph\n";
62         }
63         if (!foundGoal) {
64             std::cout << "[ERROR] Goal location {" << goal_loc.name
65                 << "} does not exist in graph\n";
66         }
67     }
68 }
```

问题 4 输出 调试控制台 终端 端口 GITLENS SPELL CHECKER 4

```
[DEBUG] 测试测试
nodes read
Error: unable to find safe starting state: tried to find t-405B at time t=0
PS E:\Company_work\delay-replanning> & 'c:\Users\14186\.vscode\extensions\ms-vscode.cpptools-1.22.11-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-jxv5uwxr.fwh' '--stdout=Microsoft-MIEngine-Out-syty0fzz.rtr' '--stderr=Microsoft-MIEngine-Error-uccqi20h.34l' '--pid=Microsoft-MIEngine-Pid-1ygy3mb3.upi' '--dbgExe=D:\msys64\mingw64\bin\gdb.exe' '--interpreter=mi'
[DEBUG] 测试测试
nodes read
Error: unable to find safe starting state: tried to find t-405B at time t=0
PS E:\Company_work\delay-replanning> & 'c:\Users\14186\.vscode\extensions\ms-vscode.cpptools-1.22.11-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-5j1lz1lc.efp' '--stdout=Microsoft-MIEngine-Out-2msmqbe3.3uc' '--stderr=Microsoft-MIEngine-Error-dvbcjn1c.ftz' '--pid=Microsoft-MIEngine-Pid-iuhx14y3.d5u' '--dbgExe=D:\msys64\mingw64\bin\gdb.exe' '--interpreter=mi'
[DEBUG] 测试测试
PS E:\Company_work\delay-replanning>
```

powershell search
cppdbg: atsipp.exe
Meson Compile (Debug) 任务

