2. **Debug** C/C++ by using gdb in VScode

- 2.1 **Install "gdb" (the debug tool of C/C++)**
  - ➢ using cmd "**which gdb**" to check whether gdb is installed or no
    - ✓ if there is no info about gbd after running command "**which gdb**", it means that gdb is not installed, then
      - 1. using "**sudo apt undate**" to update package list
      - 2. using "**sudo apt install gdb**" to install gdb

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$ which gcc
/usr/bin/gcc
ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$ which g++
/usr/bin/g++
ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$ which gdb
ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$
```

    - ✓ If the installation directory of gdb is displayed after running command "**which gdb**" is executed, it means that gdb has been successfully installed.

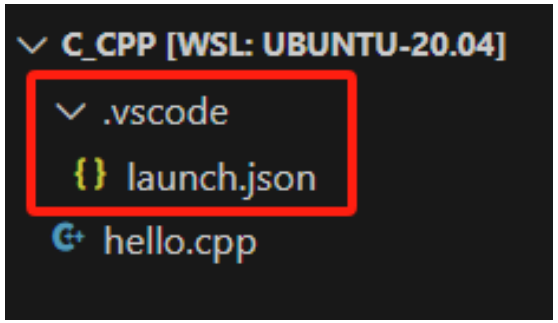
```
ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$ which gdb
/usr/bin/gdb
```

commands for install gdb

```
ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$ sudo apt update
[sudo] password for ww2:
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [3778 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [3396 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [496 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [3406 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-en [477 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [1032 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [219 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal-updates/main Translation-en [576 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [3553 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [497 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1254 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [301 kB]
Fetched 19.4 MB in 6min 8s (52.7 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
165 packages can be upgraded. Run 'apt list --upgradable' to see them.
ww2@DESKTOP-S3ETMIR:/mnt/d/2025/c_cpp$ sudo apt install gdb
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  gdbserver libbabeltrace1 libc-dev-bin libc6 libc6-dbg libc6-dev libdw1 libelf1
Suggested packages:
  gdb-doc glibc-doc
The following NEW packages will be installed:
```

2. **Debug** C/C++ by using gdb in VScode

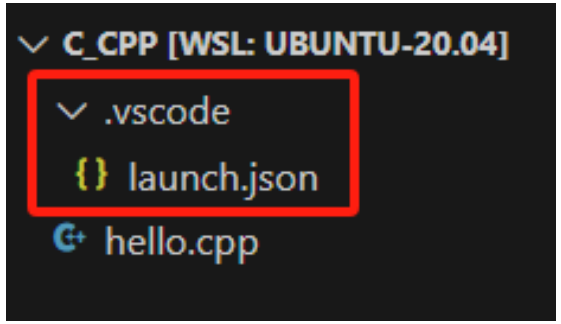
- 2.2 configure VSCode for using gdb to debug C/C++ code
  - ➢ create and edit ".vscode" folder and json files
    - ✓ step1. create a new folder named ".vscode" in the directory of C/C++ codes
    - ✓ step2. create a new json file named "launch.json" in the ".vscode" folder which is created in step1
      - edit "launch.json" to set gdb for debugging the execute file which is created by "g++ -g" / "gcc -g"
        - tips: option "-g" used with gcc/g++ is to generate information for debugging while compiling the C/C++ source code.



https://code.visualstudio.com/docs/cpp/config-linux

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(gdb) Launch",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}/${fileBasenameNoExtension}",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        },
        {
          "description": "Set Disassembly Flavor to Intel",
          "text": "-gdb-set disassembly-flavor intel",
          "ignoreFailures": true
        }
      ]
    }
  ]
}
```

<--- An example of launch.json



https://code.visualstudio.com/docs/cpp/config-linux

2. **Debug** C/C++ by using gdb in VScode

- 2.3 lunch gdb to debug in VS Code by “Run and Debug”
  - compile the souce code with “**-g**” option to generate information for debug and generate the executable file

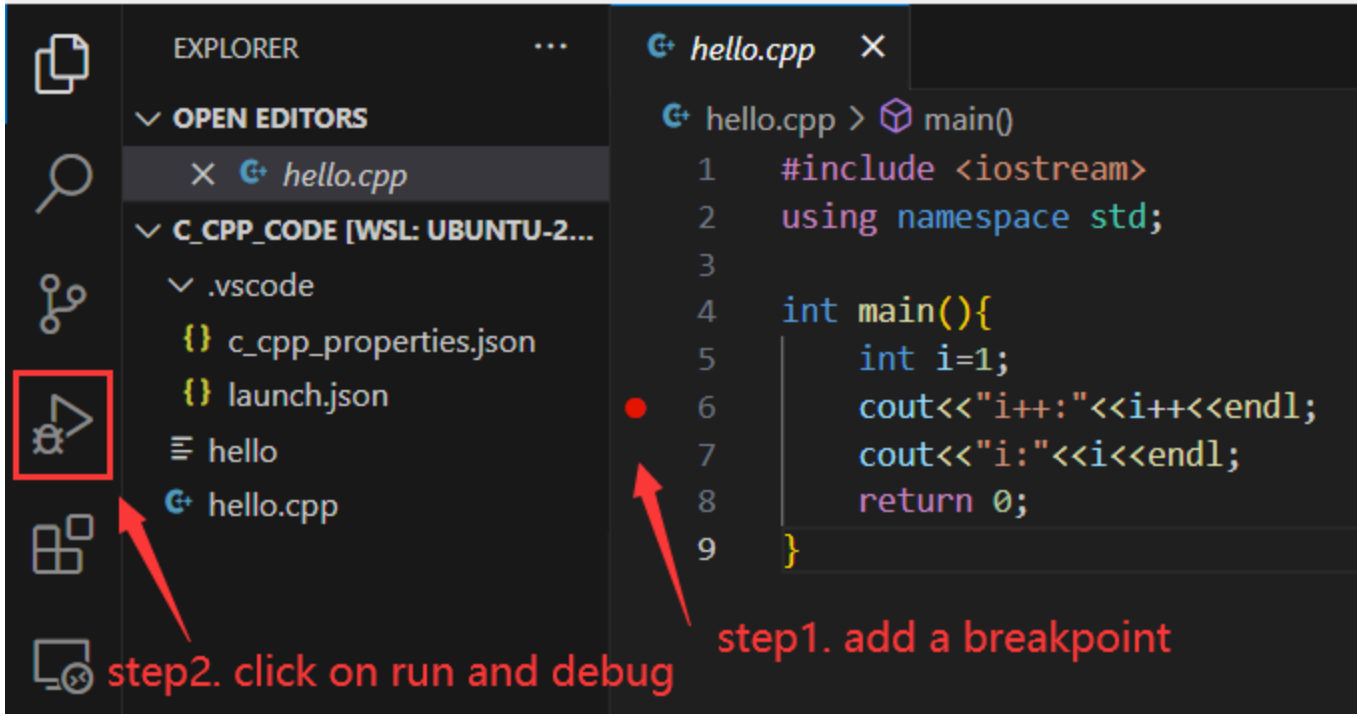
```
ww2@DESKTOP-4NIH4UK:/mnt/c/Users/sustech/Desktop/C_CPP_CODE$ ls -a
.  ..  .vscode  hello.cpp
ww2@DESKTOP-4NIH4UK:/mnt/c/Users/sustech/Desktop/C_CPP_CODE$ g++ -g -o hello hello.cpp
ww2@DESKTOP-4NIH4UK:/mnt/c/Users/sustech/Desktop/C_CPP_CODE$ ls -a
.  ..  .vscode  hello  hello.cpp
ww2@DESKTOP-4NIH4UK:/mnt/c/Users/sustech/Desktop/C_CPP_CODE$
```

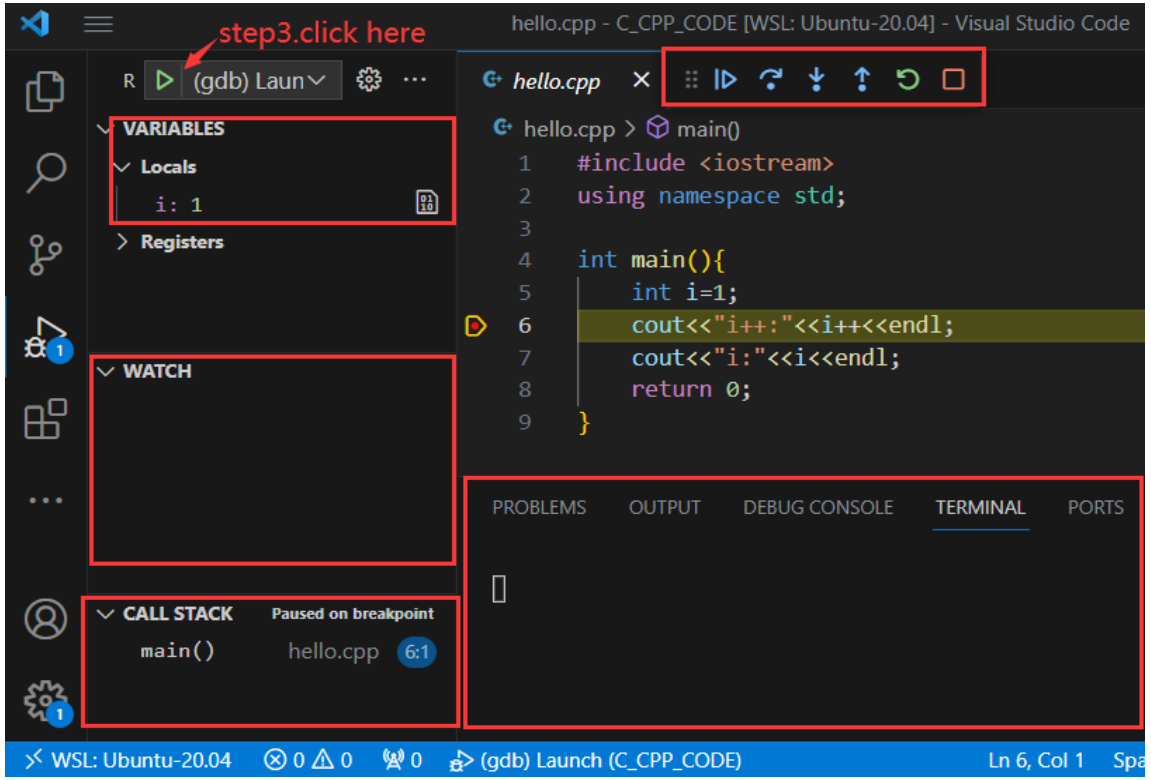
An example of launch.json-->

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(gdb) Launch",
      "type": "cppdbg",
      "request": "launch",
      "program": "${fileDirname}/${fileBasenameNoExtension}",
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
```

2. **Debug** C/C++ by using gdb in VScode

- 2.4 set “breakpoint” on source file, lunch gdb to run and debug





https://code.visualstudio.com/docs/cpp/config-linux

2. **Debug** C/C++ by using gdb in VScode

- 2.5 View the data stored in a variable by gdb(optional)
  - ➢ During debugging, you can use GDB commands to view the data stored in variable(s).

    ✓ step1. choose “DEBUG CONSOLE” window.

    ✓ step2. run the command in command line in the “DEBUG CONSOLE” window.
    - -exec [gdb command] in vscode

    ✓ step3. View the results after executing the command in the “DEBUG CONSOLE” window.

2. **Debug** C/C++ by using **gdb** in VScode

- Using the command x (for “examine”) to examine memory in any of several formats, independently of your program’s data types.
  - ✓ x /nfu addr
    - n, the repeat count
    - f, the display format
    - u, the unit size

https://sourceware.org/gdb/current/onlinedocs/gdb.html/Memory.html#Memory

hello.cpp > main()

```cpp
using namespace std;

int main(){
    cout <<"sizeof(char): "<<sizeof(char)<<" byte(s)"<<endl;
    char x = 0xFF;
    char y = 'b';
    char z = 'B';
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS MEMORY

```
sizeof(char): 1 byte(s)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
-exec x /1xb &x
0x7fffffffddfd: 0xff
-exec x /1tb &x
0x7fffffffddfd: 11111111
-exec x /1ob &x
0x7fffffffddfd: 0377
-exec x /1db &x
0x7fffffffddfd: -1
-exec x /1ub &x
0x7fffffffddfd: 255
-exec x /1cb &y
0x7fffffffddfe: 98 'b'
-exec x /1db &y
0x7fffffffddfe: 98
-exec x /1cb &z
0x7fffffffddff: 66 'B'
-exec x /1db &z
0x7fffffffddff: 66

> -exec x /3xb &x
```