有关编程与调试环境的使用

一些基本概念

什么是代码编辑器

顾名思义，是编辑代码的地方，理论上windows的记事本也可以叫做代码编辑器，但是它的功能太少了（linux中gedit命令可以打开类似于windows记事本的编辑器）。

真正可用的代码编辑器应该包括如下多种功能：

- **语法高亮**：代码编辑器能够识别不同的编程语言，并通过不同的颜色和字体样式来突出显示关键字、变量、函数等，这样可以提高代码的可读性。
- **代码自动完成**：很多代码编辑器都提供代码自动完成功能，可以根据上下文提示并自动完成代码，这样可以提高编码效率。
- **代码调试**：一些高级的代码编辑器集成了调试工具，允许程序员设置断点、单步执行、查看变量值等，帮助程序员找到并修正代码中的错误。
- **插件系统**：许多代码编辑器支持插件，允许用户根据需要安装额外的功能。

代码编辑器有些什么

- Visual Studio Code(VS Code)：它通过插件下载可以支持几乎所有的主流编程语言，包括Python、Java、C++等
- JetBrains IntelliJ IDEA：它主要支持Java，是集成开发环境。
- PyCharm：它专门针对Python开发，是集成开发环境。

什么是编译器

计算机读不懂人类的语言，它只能读懂二进制语言，为了方便开发代码，人类在二进制语言和人类语言中进行了折中，开发了高级语言，如C，C++等（与之对应的低级语言就是二进制语言）。

编译器就是将程序员用高级编程语言编写的源代码转换成计算机可以直接执行的机器代码或另一种更低级、更易于执行的编程语言的软件。

常见的编译器有GCC，Clang，MSVC（Microsoft Visual C++编译器）等。

什么是IDE（集成开发环境）

它为计算机程序员提供了一个综合性的工具集，用于软件开发。IDE通常将以下功能集成在一个统一的用户界面中：

- **源代码编辑器**：提供语法高亮、代码自动完成、代码折叠、错误提示等功能，以提升编码效率。
- **编译器/解释器**：将源代码转换成计算机可以执行的机器代码或者直接执行源代码。
- **调试器**：允许程序员设置断点、单步执行、查看变量值、监控程序状态等，帮助查找和修复代码中的错误。
- **构建工具**：自动化编译、链接、打包等构建过程。
- **版本控制系统**：集成Git、SVN等版本控制工具，方便代码管理和团队协作。
- **项目管理工具**：帮助管理项目文件、资源、配置等。
- **模拟器/仿真器**：对于移动或嵌入式开发，提供模拟器来测试应用程序在不同设备上的表现。
- **文档生成器**：自动生成代码文档。

什么是调试器

在写代码时，几乎不可能保证一次写对，因此需要一个软件来让我们知道执行过程中的具体情况，这就是调试器。

调试器通常提供以下功能：

- **断点设置**：允许程序员在代码的特定位置设置断点，当程序执行到这些位置时，调试器会暂停程序的执行。
- **单步执行**：程序员可以逐行或逐指令地执行程序，观察程序的状态和行为。
- **变量监视**：调试器可以显示当前作用域中所有变量的值，以及它们的类型和属性。
- **调用栈查看**：调试器可以显示当前的函数调用栈，帮助程序员理解程序的执行流程。
- **条件断点**：可以设置条件断点，只有当指定的条件满足时，程序才会暂停。
- **表达式求值**：在调试过程中，程序员可以在调试器中计算和修改程序中的表达式。
- **内存检查**：调试器允许程序员检查和修改内存中的数据。

常见的调试器有gdb、LLDB、Visual Studio Debugger等。

安装运行环境

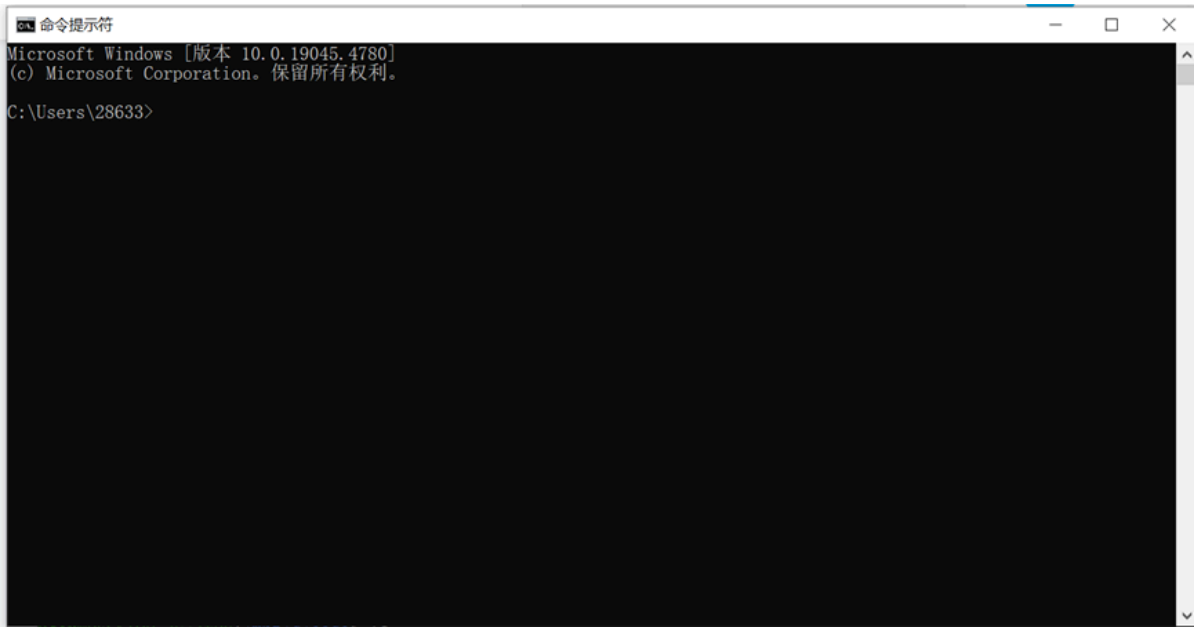
我们提供两种运行环境，一种是WSL，另一种是虚拟机。

WSL的安装

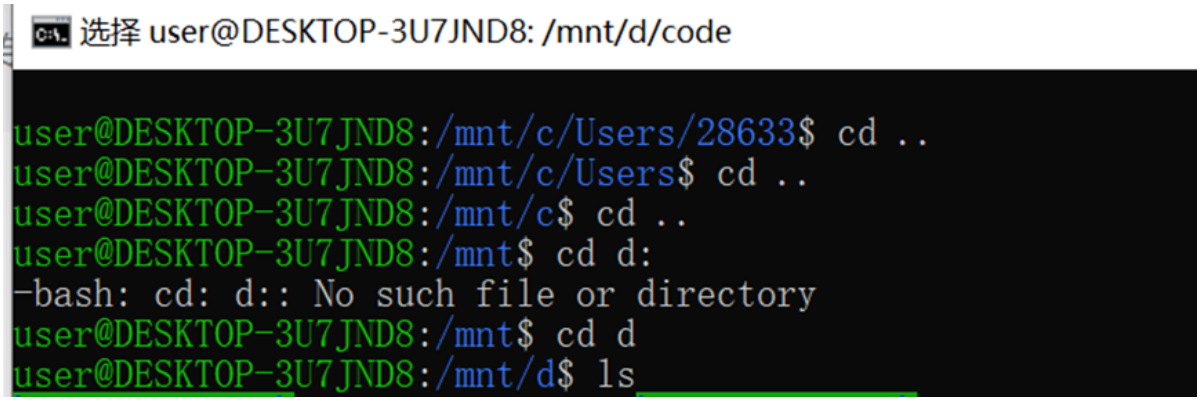
Windows下也存在一种管理Linux文件的方式，称作WSL（Windows Subsystem for Linux），Windows 10\11上能够运行原生Linux二进制可执行文件（ELF格式）的兼容层。它是由微软与Canonical公司合作开发，开发人员可以在 Windows 计算机上同时访问 Windows 和 Linux 的强大功能。 通过适用于 Linux 的 Windows 子系统，开发人员可以安装 Linux 发行版（例如 Ubuntu OpenSUSE、Kali、Debian、Arch Linux 等），并直接在 Windows 上使用 Linux 应用程序、实用程序和 Bash 命令行工具，不用进行任何修改，也无需承担传统虚拟机或双启动设置的费用。

它的下载安装方式可以参考这个帖子。[如何开启win10内置Linux子程序 - 哔哩哔哩 (bilibili.com)]

在安装完它以后，可以通过windows下方搜索栏输入cmd先进入windows的命令行界面，如下图；然后再输入wsl。



最终效果如图：



这里的mnt以后的东西就是windows上面的文件系统，可以考虑在/mnt/d里面新建一个code文件夹，然后在里面编写代码，同时，windows下就可以看到这个改动。

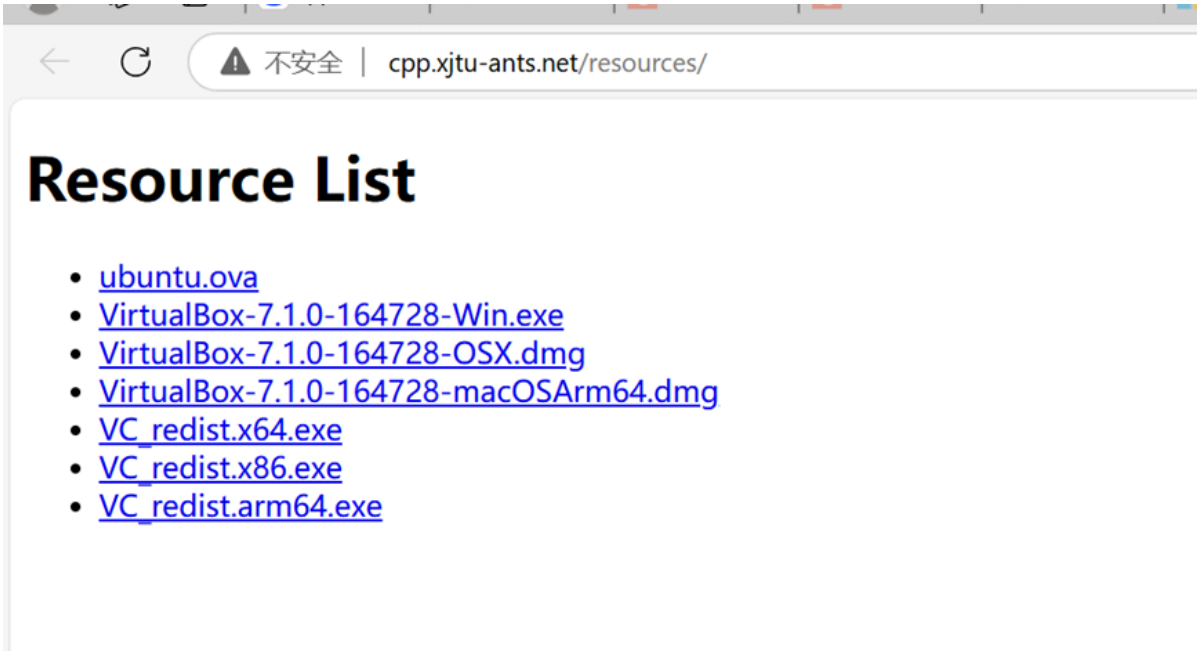
虚拟机的安装

Windows管理虚拟机的软件一般为VMware或者Virtual Box。

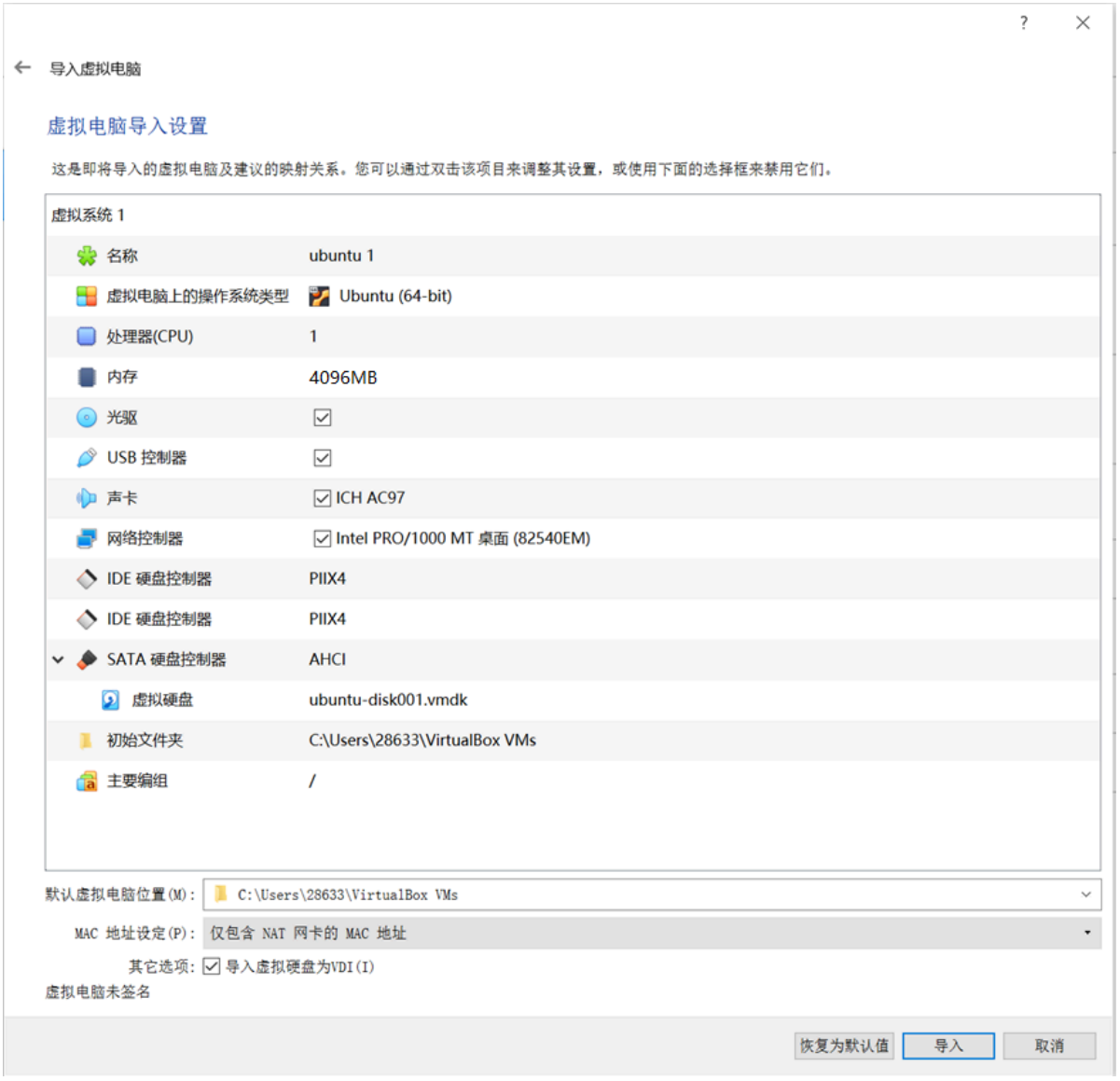
为了方便操作，已经将virtualbox的虚拟机放在了学校的服务器上，需要同学们下载然后在virtualbox中导入虚拟机。也可以自行尝试按照网上教程安装。

具体下载方式为：登录浏览器，在上方输入网址：http://cpp.xjtu-ants.net/resources/

正确的界面如下：



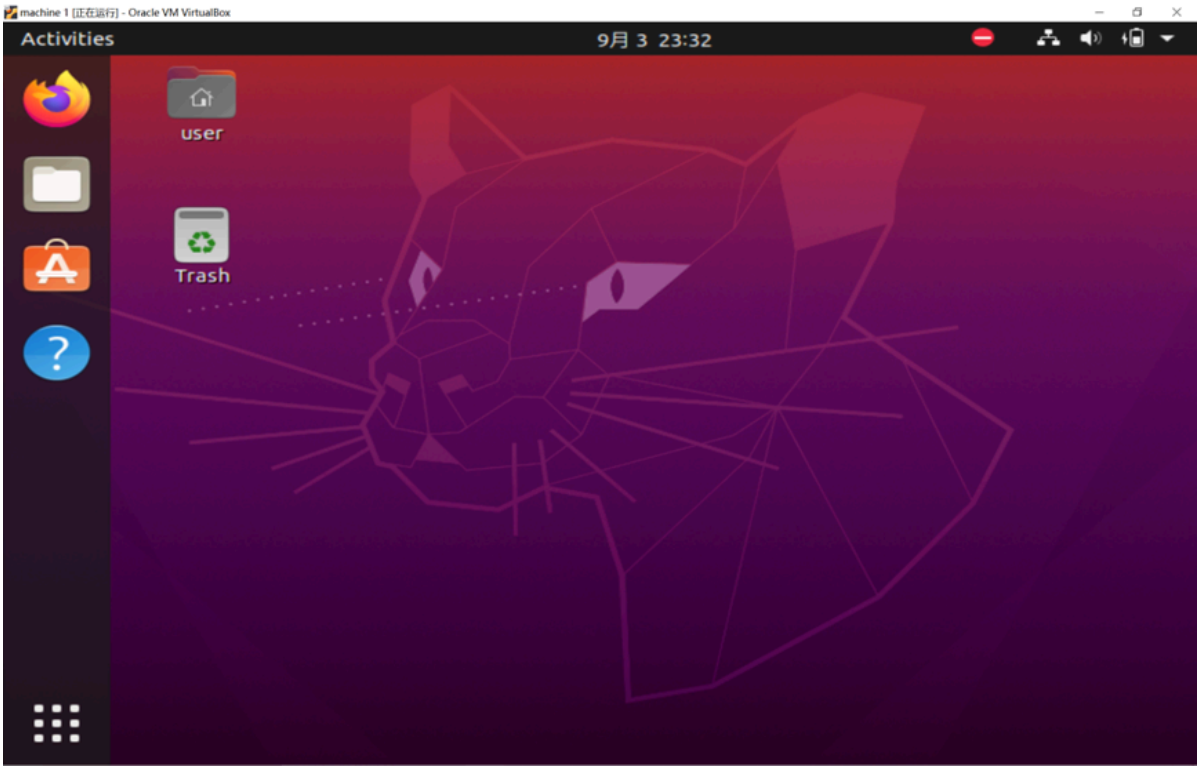
将这两个文件下载，第二个是mac的virtualbox版本，第三个是windows的virtualbox版本（两个virtualbox版本均为6.1.50），点击下载virtualbox；第一个是打包好的虚拟机，需要将它导入virtualbox中，具体导入方式为：点击管理中的“导入虚拟电脑”，然后选择服务器中的ova文件。



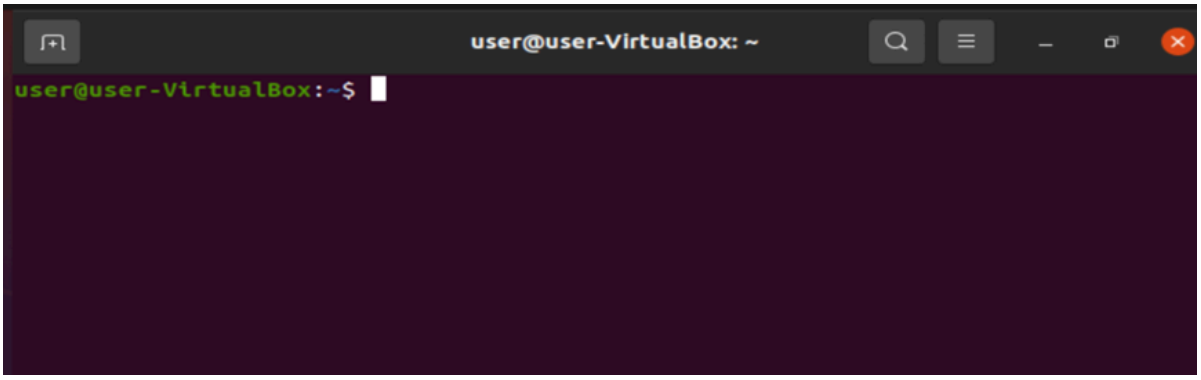
可以选择修改默认的虚拟电脑位置，修改好后点击“导入”即可。

导入完成后，点击“启动”将虚拟机启动。

虚拟机的用户名：user，密码：123。在进入了虚拟机后，正常界面是这样的：



此时可以使用`Ctrl+Alt+t`的快捷键来打开终端，也可以在左下方的省略号里面找到terminal打开。终端是这样的：



Linux 基本知识

下载软件

一般情况下，可以用`apt-get install xxx`来下载指定软件，如果权限不够，则可以在前面加上`sudo`来提升权限。

这个环境内尚且有不少功能没有下载，需要运行以下命令：

```
sudo apt-get update
sudo apt-get upgrade
sudo apt install vim
sudo apt install g++
```

这样做的目的是更新依赖，安装此次任务必须的功能。

代码编辑器

可以使用vim，vim即为linux原生编辑软件；如果不习惯，开始可以选用`gedit`来打开类似windows的记事本一样的编辑器。

一部分Linux终端基础指令

- `cd`指令用于进入一个文件夹，如当前文件夹下一个文件夹名为folder，则指令为cd folder
- `cd ..`用于返回上级文件夹。
- `mkdir`用于新建文件夹；如想新建一个folder的文件夹，则指令为`mkdir folder`
- `ls`用于查看文件夹中的东西；如果想看当前文件夹下的内容，则指令直接为`ls`
- `man`指令可以查看一个指令的用法，比如`man ls`

写一个代码并测试

使用vim

上面提到vim是一款文本编辑器，在Linux中自带，它是一个简约但是具有高可定制性的编辑器。我们刻意使用命令行写代码并进行测试，为的是让大家了解并入门命令行工具。

打开命令行界面，新建一个文件夹，然后创建一个新的C++代码文件：

```
mkdir test
cd ./test
vim test.cpp
```

进入vim以后，可以发现什么都没有（因为打开的是一个新文件），此时的vim处在**普通模式**，这时候可以进行删除、复制、粘贴操作，下面给出一些常用的按键功能（不要求掌握）：

- `jkhl`可以移动光标指针，上下左右键也一样
- `gg`可以移动至文件开头，`G`可以移动至文件尾
- `v`从光标开始选中，此时移动光标即可选定选中范围，按`y`拷贝至剪贴板，粘贴则是`p`（粘贴到光标之后，可能会在实际中略有出入）
- `x`删除光标所指字符，`dd`删除所在行（用不习惯的用`back space`也差不多）

普通模式按`i`（即insert）进入插入模式，此时可以通过键盘输入将代码写进去。

插入模式按`ESC`可以回到普通模式。

普通模式按`:`进入末行模式。末行模式主要用于保存退出。

末行模式一般输入`wq`并按回车即可保存代码，`w`的含义是保存，`q`的含义则是退出（quit），可以分开单独使用。可以添加`!`代表强制执行（override）。

这里写一个简单的计算和式的代码，目的是为了便于使用gdb调试（请注意不要压行，即刻意把若干行代码写到一行内）：

打开vim按`i`，输入以下代码：

```
#include <iostream>
using namespace std;
int main() {
    int sum = 0;
    for(int i = 1; i <= 100; ++i){
        sum = sum + i;
    }
    cout << sum << endl;
    return 0;
}
```

按`ESC`后按`:`，输入`wq`按回车，保存文件并退回到命令行。

编译你的代码

这里使用g++来编译刚刚写好的程序，在终端内输入：

```
g++ -g -o test.exe ./test.cpp
```

- `g++`是应用的名称，即运行GNU C++编译器，后面都是它的参数
- `-g`的含义是：输出的可执行文件是能够被调试的（否则无法在gdb内正常运行）
- `-o test.exe`的含义是：输出的可执行文件名是`test.exe`，在不指定它的时候默认输出是`a.out`
- `./test.cpp`即刚写好的源代码

看一看子目录里面新增了什么：

```
user@user:~/test$ ls
test.cpp  test.exe
```

编译好的结果就出现在了目录下方。

运行一下看看：

```
user@user:~/test$ ./test.exe
5050
```

调试你的代码

当代码运行与预期结果不一致时，很可能就需要逐行运行代码，定位问题所在。

我们使用gdb调试代码，在终端内输入：

```
gdb ./test.exe
```

此时进入调试界面，此时可以在某些行处设置断点，即程序运行到断点所在行时，会被暂停并移交控制权给用户，这里在第4行处打一个断点，输入`b 4`：

```
(gdb) b 4
Breakpoint 1 at 0x11b5: file ./test.cpp, line 4.
```

然后输入`run`开始运行程序，程序运行到断点时自动停止，显示这一行的内容，此时这一行尚未被执行：

```
(gdb) run
Starting program: /home/user/test/test.exe
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at ./test.cpp:4
4       int sum = 0;
(gdb)
```

按`n`（即`next`，执行下一行。也可以按`s`（`step`，单步执行），区别是前者会执行完整行语句，后者会进入到函数（如果有）内）可以执行这一行的内容，或者按`Enter`运行上一个非`Enter`的命令（例如按一下`n`后，一直按`Enter`就会一直一行一行地执行代码），按几次以后，输入`p 变量名`查看变量的值（`p`即`print`。后面的变量名也可以替换成表达式），根据按的次数不一样看到的结果也可能不一样：

```
(gdb) p sum
$3 = 15
(gdb) p i
$4 = 5
```

输入`c`（即`comtinue`，从当前行开始，一直运行程序）让它运行到最后吧：

```
(gdb) c
Continuing.
5050
[Inferior 1 (process 1703) exited normally]
```

输入`quit`就可以退出了。

当然了，gdb功能远比这个强大，你可以在运行时跟踪某个变量，修改某个变量的值，查看栈帧等等。

参考资料

- 此资料设计动机：计算机教育中缺失的一课：计算机教育中缺失的一课
- WSL的安装：安装 WSL | Microsoft Learn
- Ubuntu20.04桌面版下载链接（若需要自行安装则可以访问）：https://mirrors.tuna.tsinghua.edu.cn/ubuntu-releases/20.04/ubuntu-20.04.6-desktop-amd64.iso
- 常用Linux指令及其用法：工作中总结的30个常用Linux指令，实在记不住就别硬记了，看这篇就够了 - JavaBuild - 博客园 (cnblogs.com)
- 较全面的vim操作指南：vim简单使用教程 - 走在大牛的路上 - 博客园 (cnblogs.com)
- 常用g++参数：g++重要编译参数 - 北极星！ - 博客园 (cnblogs.com)
- gdb调试命令一览：linux下gdb调试方法与技巧整理_gdb调试c++-CSDN博客