编译、运行的基础知识

如果你是一个 CS 专业的学生，你应当已经对此有所了解，并且尽可能知晓这里的每一个细节。

首先我们需要明确的是，**编译**一个 C++ 程序是由**编译器**来完成的，我们通过在**命令行（终端）** 调用编译器并传递相应的参数，来告知编译器我们想要编译哪个文件、采用什么标准、什么警告级别、生成的可执行文件的名字等等。一切 IDE 为你提供的"一键编译运行"功能都只是在背后做了这件事。 **VSCode, CLion 之类的 IDE 并没有编译、运行的能力！**

例如，

```
g++ hello.cpp -o hello -std=c++20
```

这行指令会调用 C++ 的编译器 `g++` ，要求它编译那个名为 `hello.cpp` 的文件，生成的可执行文件叫做 `hello` （在 Windows 上叫 `hello.exe`），采用 C++20 标准。其中，终端会在环境变量 `PATH` 所保存的目录里找到名为 `g++` 的可执行文件（在 Windows 上叫 `g++.exe`），这就是为什么你在安装编译器的时候需要将它的 `bin` 目录添加到环境变量 `PATH` 。

终端有所谓 working directory 的概念：如果你在 Windows 上启动一个 PowerShell ，你会看到类似于这样的提示

```
C:\Users\gkxx>
```

伴随一个光标在 `>` 后面。这里的 `C:\Users\gkxx` 就是当前的 working directory 。注意， Windows 习惯用 `\` 作为路径分隔符，而 Linux 和 Mac 用 `/` 。考虑到绝大多数同学的系统都是 Windows ，以下我都使用 `\` 。我们可以通过 `cd` 指令来改变 working directory ：

```
C:\Users\gkxx> cd D:\cs101\pa3
D:\cs101\pa3>
```

如果你使用 VSCode ，你可以按 ctrl+`` ` `` （键 `1` 左边的那个）启动一个终端，它的默认 working directory 是当前 VSCode 打开的文件夹。你还可以在 `.vscode\settings.json` 中设置 `"terminal.integrated.cwd": "my\desired\working\directory"` 来将默认 working directory 修改为 `my\desired\working\directory` 。

当我们执行 `g++ hello.cpp -o hello` 的时候， `hello.cpp` 指的就是当前 working directory 下的 `hello.cpp` ， `-o hello` 也是要求在当前 working directory 下创建可执行文件。

有时候，我们会需要向编译器传递相对路径。假如在 `pa3` 目录下有一个名为 `tmp` 的文件夹，里面存放着我们想要编译的代码文件 `hello.cpp` ，我们可以

```
D:\cs101\pa3> g++ tmp\hello.cpp -o tmp\hello
```

这时编译器就编译了 `tmp` 文件夹下的 `hello.cpp` ，并将生成的可执行文件也存放在 `tmp` 里。以上两个相对路径都可以任意更改，只要所指涉的文件和路径存在即可。

在相对路径中，我们还可以使用 `.` 来表达当前目录， `..` 表示上一级目录。例如，下面的指令

```
D:\cs101\pa3> g++ ..\pa2\prob2\main.cpp -o .\tmp\main_from_pa2
```

编译了 `D:\cs101\pa2\prob2\main.cpp` ，并生成名为 `main_from_pa2.exe` 的可执行文件，存放在 `D:\cs101\pa3\tmp` 下。

要运行一个可执行文件，只需输入它的相对路径，并且可以不写 `.exe` ；但如果它就在当前 working directory 下，你需要在开头加上 `.\` 。如果 `hello.exe` 是一个输出 `"Hello, world"` 的程序，在不同位置运行它的效果如下：

```
D:\cs101\pa3> tmp\hello
Hello, world
D:\cs101\pa3> cd tmp
D:\cs101\pa3\tmp> .\hello
Hello, world
```