Windows10和Linux下运行C,Fortran等程序若干注意事项

作者：徐黎闽

单位：清华大学数学系

如果仅限于一些数学上的研究，那使用 **Matlab** 和 **Python** 就足够了.

但有些时候，我们不仅限于数学上研究的需要，我们还可能需要去编写并运行一些C，C++，Fortran程序（比如**第一性原理计算软件**的）。这就涉及是在Windows10还是在Linux下操作了。

我一般遵循下面的原则：

- 如果是编写小型的测试文件（不需要Lapack，FFTW等外部库），则选择在Windows10下用Vscode编写。
- 如果是编写稍大一点的较正式文件或者需要一些外部库之类的文件，则选择在Linux下用Vim编写。

原因是：只要将Vscode配置好，那运行小型的测试文件是很方便的；但如果文件需要依赖于外部库，那想在Windows上配置好这些库比较麻烦。再者，大型的Fortran文件的天然土壤就是Linux。

下面分Windows还是Linux分别讲解相关注意事项。

1.Windows10中运行

编写一个程序并运行，需要几个工具：

- 编辑器：用来编写程序代码
- 编译器：用来编译程序代码，将之转换为二进制语言，可以直接运行

对于前者，理论上，可以用任何写入文本的工具作为编辑器，在Windows下我们使用Vscode。

对于后者，我们的编译器应该对应于我们具体的语言。即C语言应该用C语言的编译器来编译，Fortran语言应该用Fortran语言的编译器来编译。这里我们使用 mingw-w64。它是C/C++/Fortran编译器 gcc 的windows版本。

有了这两个后，我们就可以编写C/C++/Fortran程序并运行了。但是，为了编写程序的方便，我们还需要在Vscode中下载插件 **C/C++** 和 **Modern Fortran**, 这两个插件分别可以帮助我们进行 C/C++ 和 Fortran 的 **代码补全** 和 **程序调试** 等。

目的：**运行C，C++，Fortran等语言**

工具： **1. Vscode 作为编辑器；2. mingw-w64 作为编译器；3. C/C++插件和Modern Fortran插件**

如果我们的程序只包含一个或两个文件，那我们可以直接命令行输入编译命令；但如果有多个，命令行输入就不方便了。

此时我们就需要编写 **Makefile** 文件，Makefile文件是将多个文件编译时的依赖关系在文件中指明。然后我们在Makefile文件所在文件夹的目录处输入**make** 命令，就可以完成编译了。

在windows10下，由于我们使用的是mingw-w64,所以自带的命令是**mingw-w64-make**;为了方便，我们可以**将mingw-w64-make 程序重命名为 make 程序**，这样我们就可以直接输入 **make** 命令了。（如果不这么做的话，Vscode终端下输入make就会报错说找不到命令，只能输入 mingw-w64-make 命令）

1.1 make 举例

my.h头文件

```
#ifndef _MY_H
#define _MY_H
int sum(int m, int n);

#endif
```

my.c文件，函数实现

```
#include"my.h"
int sum(int m, int n) {
    int i, sum = 0;
    for (i = m; i <= n; i++) {
        sum += i;
    }
    return sum;
}
```

main.c文件，主函数

```
#include <stdio.h>
#include "my.h"

int main()
{
    printf("%d\n", sum(1, 1000));
    return 0;
}


```

makefile文件

```
main:main.o my.o
    gcc main.o my.o -o main
main.o:main.c
    gcc -c main.c
my.o:my.c
    gcc -c my.c
.PHONY:clean

#linux 下用 rm -rf *.o main
clean:
    @echo "=======clean project========="
    del  *.o
    @echo "=======clean completed========="
```

输入命令后显示情况：

```
PS F:\code\study_c\multi> make
gcc -c main.c
gcc -c my.c
gcc main.o my.o -o main
PS F:\code\study_c\multi> ./main
500500
PS F:\code\study_c\multi> make clean
"=======clean project========="
del  *.o
"=======clean completed========="
```

1.2 mingw-w64安装

步骤如下：

- 去官网下载，放在D盘（我习惯放在D盘）
- 安装（**记住安装目录**）
- 将 **安装目录** 添加到 **系统的环境变量**：1.右键 “此电脑”；2.点击 “高级系统设置”；3. 点击 “环境变量”；4. 点击 ”系统变量“；5. 点击 ”Path“；6. 点击 ”编辑“；7.点击 ”新建“，将刚才的安装目录复制到新建里即可。

2.Linux中运行

Linux下也一样，我们需要：

- 编辑器：我们选择 Vim
- 编译器：一般有 GNU 和 Intel 两个常用的编译器，我们选择GNU编译器。

GNU编译器和 Intel 编译器的使用命令是不一样的：

- **GNU：gcc（C/C++），gfortran（Fortran）**
- **Intel：icc（C/C++），ifort（Fortran）**

我们使用的Linux系统是Ubuntu，这个Linux自带的编辑器是Vi，特别不好用，我们把Vi换成Vim.下载安装好 Vim后，我们再下载安装 GNU编译器。

对于数学家和物理学家来说，使用Linux的目的是进行 **科学计算**！因此，自然经常使用 **Lapack库** 和 **FFTW库**。所以我们还需要安装Lapack库 和 FFTW库。

最后需要注意的一点是，我们在Windows10下mingw-w64自带make，同样，在GNU中也自带make。但是由于我们只用Linux的目的是针对大型的需要依赖外部库的程序，对于这样的程序来说，有些时候编写makefile也是一件困难的事。因此，我们就需要 cmake。（更重要的是，Lapack的安装是基于 cmake 的；所以就算我们不安装 cmake 也不行）

总结起来如下：

- **下载安装编辑器 Vim**
- **下载安装编译器GNU（带有C/C++/Fortran编译器和 make）**
- **下载 cmake （Lapack的安装基于这个）**
- **下载安装Lapack库（包含线性代数、数值积分等）**
- **下载安装FFTW库（包含快速傅里叶变换等）**

2.1 cmake 介绍

makefile在一些简单的工程完全可以人工手下，但是当工程非常大的时候，手写makefile也是非常麻烦的，如果换了个平台makefile又要重新修改。

这时候就出现了Cmake这个工具，cmake就可以更加简单的生成makefile文件给make用。当然cmake还有其他功能，就是可以跨平台生成对应平台能用的makefile，你不用再自己去修改了。（即**makefile自身不具有平台移植性，但是cmake可以生成对应平台的makefile，从而使用cmake具有平台移植性**）

可是cmake根据什么生成makefile呢？它又要根据一个叫**CMakeLists.txt**文件去生成makefile。到最后CMakeLists.txt文件谁写啊？你自己手写！当然如果你用IDE（集成开发环境），类似VS这些一般它都能帮你弄好了，你只需要按一下那个三角形。

2.2 安装编辑器 Vim

这里以虚拟机为例.

首先检查网络是否畅通(ping 一下网络 ip地址)，**禁用共享文件夹**，执行 `sudo apt-get install vim`.

此时很可能出现 **无法连接上cn.archive.ubuntu.com软件源** 的问题，这是由于这个软件源出了问题。我们的解决办法就是更换软件源：

- 输入 `sudo vim /etc/apt/sources.list`
- 将 `sources.list` 中的 `cn.archive.ubuntu.com` 和 `security.archive.ubuntu.com` 全部更换为 可用的源（如：`mirrors.aliyun.com`）可以去网站源列表 - Ubuntu中文查看可用的软件源。
- 刷新一下 `sudo apt-get update`，这是由于我们刚才只是修改了sources.list，只有执行了刷新，系统才知道sources.list修改了，才能起作用。

2.3 安装编译器 GNU

(1) gcc

Ubuntu下自带gcc编译器，可通过 `gcc -v` 查看.

(2) g++

命令： `sudo apt-get install build-essential` .执行完后，就完成了 gcc,g++,make的安装. build-essential 是一整套工具，gcc,libc等。通过 `g++ -v` 查看.

(3) gfortran

命令：`sudo apt-get install gfortran` .通过 `gfortran -v` 查看.

2.4 安装 cmake

CMake是开源、跨平台的构建工具，可以让我们通过编写简单的配置文件去生成本地的Makefile，这个配置文件是**独立于运行平台和编译器**的，这样就不用亲自去编写Makefile了，而且配置文件可以直接拿到其它平台上使用，无需修改，非常方便。

命令：`sudo apt install cmake` .通过 `cmake -version` 查看.

使用时，在主文件所在目录下输入命令：`cmake .`

2.5 安装Lapack库

步骤如下：

- 下载Lapack库（可以直接下载好后放在家目录下 / 使用命令 `wget http://www.netlib.org/lapack/lapack-3.5.0.tgz`）
- 解压 `tar -zxvf lapack-3.5.0.tgz`，并进入目录里 `cd lapack-3.5.0/`
- 最简单的方法 `sudo apt-get install libblas-dev liblapack-dev`，就按照好了.

注意1：Ubuntu下如果没有指定安装目录的话，会默认安装在 `/usr/local/lib` 下.

注意2：除了上面的最简单的方法外，还有使用cmake的方法，如下

```
cd lapack-3.9.0/
mkdir build
cd build
cp ../make.inc.example make.inc
cmake .

sudo cmake --build . --target install
...
[100%] Built target lapack
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/lib/cmake/lapack-3.9.0/lapack-targets.cmake
-- Installing: /usr/local/lib/cmake/lapack-3.9.0/lapack-targets-
release.cmake
-- Installing: /usr/local/lib/pkgconfig/lapack.pc
-- Installing: /usr/local/lib/cmake/lapack-3.9.0/lapack-config.cmake
-- Installing: /usr/local/lib/cmake/lapack-3.9.0/lapack-config-version.cmake
-- Installing: /usr/local/lib/pkgconfig/blas.pc
-- Installing: /usr/local/lib/libblas.a
-- Installing: /usr/local/lib/liblapack.a
```

2.6 安装FFTW库

源码安装一般都遵循三个步骤：

- **configure**
- **make**
- **make install**

FFTW安装步骤如下：

- 去官网下载FFTW安装包，放在家目录下
- 解压：`tar -zxvf fftw-3.3.6-pl2.tar.gz`
- 进入文件夹：`cd fftw-3.3.6-pl2`
- ```
  ./configure
  make
  make install
  ```
  ```

这样就安装结束了。这样FFTW将会安装到默认的路径下，由于我们使用的是GNU编译器，所以默认安装在 `/usr/local/lib` 下.

有时候，我们会根据 `./configure -help` 得到的信息来添加一些参数来达到优化编译的目的.

下面给出常用的配置参数：

```
Installation directories:

--prefix=PREFIX    install architecture-independent files in
PREFIX[/usr/local]

设定安装目录



Optional Features:

--enable-shared[=PKGS]    build shared libraries [default=no]

是否编译动态库



--enable-static[=PKGS]    build static libraries [default=yes]

是否编译静态库



--enable-single         compile fftw in single precision

--enable-float          synonym for --enable-single

是否编译单精度版本



--enable-sse            enable SSE optimizations

--enable-sse2           enable SSE/SSE2 optimizations

--enable-avx            enable AVX optimizations

--enable-avx2           enable AVX2 optimizations

--enable-neon           enable ARM NEON optimizations

开启针对特定机器架构的优化，这个取决于机器CPU（下面有介绍）。



--enable-fma            enable optimizations for machineswith fused
multiply-add

开启积和熔加运算(Fused Multiply-Add/FMA)的优化

```

```
--enable-mpi            compile FFTW MPI library

是否编译mpi版的fftw库



--enable-openmp         use OpenMP directives for parallelism

是否使用OpenMP指令进行并行



--enable-threads        compile FFTW SMP threads library

是否编译FFTW SMP线程库



Some influential environment variables:

  CC          C compiler command

  CFLAGS      C compiler flags

  CPP         C preprocessor

  MPICC       MPI C compiler command

  F77         Fortran 77 compiler command

  FFLAGS      Fortran 77 compiler flags

这部分是指定编译器及编译参数，默认是用GNU的编译器。

为了用intel的编译器，我们需要特别指定一下：CC=icc F77=ifort，相关的参数通常保持默认即可。
```

举个例子如下:

```
tar zxvf fftw-3.3.3.tar.gz
cd fftw-3.3.3

#单精度

./configure --prefix=$HOME/software/fftw/3.3.6-pl2-icc13 \
  CC=icc F77=ifort \
  --enable-shared --enable-static \
  --enable-float \
  --enable-sse --enable-sse2 \
  --enable-avx --enable-avx2 --enable-fma \
  --enable-mpi \
  --enable-threads--enable-openmp
make
make install

#clean一下

```

```
make clean

#双精度

./configure --prefix=$HOME/software/fftw/3.3.6-pl2-icc13 \
  CC=icc F77=ifort \
  --enable-shared --enable-static \
  --enable-sse2 --enable-avx --enable-avx2 --enable-fma \
  --enable-mpi \
  --enable-threads--enable-openmp
make
make install

```

2.7 Lapack和FFTW使用的注意事项

这里，我们要提一下 Lapack和FFTW 使用时的注意事项：

- 由于FFTW3和FFTW2相比，做了较多的改动，所以导致一些命令是不能共用的。因此在装了FFTW3的系统上运行使用FFTW2的程序几乎一定会报错。解决这些报错只能通过仔细阅读FFTW3的官方文档来解决。
- 由于Lapack不同的版本安装结束后得到的文件名并不相同，所以在别的系统上运行正确的程序可能在你的系统上make后报错出问题。解决的办法如下：
  - 去 `/usr/local/lib` 中查看你安装Lapack后得到的文件，我的是 `libtmglib.a` `liblapack.a` `libblas.a`
  - 打开你程序的 **makefile** 文件，把文件中依赖于 **Lapack** 这个外部库的命令检查一下，看是否都是上面那几个文件，将之全部换为上面那几个文件
  - 保存退出，make 一下即可