附录 A 设置 Fortran 开发环境

在我们深入代码之前，让我们先了解一下编辑、编译和运行 Fortran 程序的基础知识。我会推荐一些我喜欢的文本编辑器，并指导您设置完整的 Fortran 开发环境。如果您熟悉 Docker 并且想跳过所有繁琐的设置步骤，请直接跳转到第 A.4 节，在那里我会介绍如何获取现代 Fortran Dockerfile，让您立即上手。

A.1 编辑 Fortran 源文件

您将编写 Fortran 程序和模块作为纯文本文件。您可以在您喜爱的文本编辑器中编辑它们。以下是一些流行的选择：

- Vim（Vi IMproved，https://www.vim.org/）是轻量级且功能强大的编辑器，对于初学者来说有一定的学习曲线。这是我个人的选择。我在 2006 年开始编程时开始使用它，从那时起就再也没有换过其他编辑器。
- Emacs（https://www.gnu.org/software/emacs）是一个功能强大且可扩展的编辑器，也是仍然在主流使用中的最古老的应用程序之一。
- Atom（https://atom.io）是由 GitHub 开发的现代、功能丰富的集成开发环境。Atom 还具有一个内置的包管理器，可以向编辑器添加第三方功能。
- Visual Studio Code（https://code.visualstudio.com）是另一个现代的、功能齐全的集成开发环境，类似于 Atom。在选择文本编辑器时，一个重要的特性是它是否可以用不同颜色突出显示 Fortran 语法。我列出的所有编辑器都可以做到，无论是开箱即用还是通过扩展。它们还都是免费且开源的，因此如果您还没有喜欢的编辑器，我建议您尝试每一个，并看看哪一个最舒适。

Fortran 源文件的扩展名

虽然 Fortran 标准对 Fortran 源文件的扩展名没有强加任何约束，但编译器供应商已经采用了几乎通用的一套规则：

- 带有后缀 .f、.for 或 .ftn 的文件名被解释为固定格式（FORTRAN 77 及更早版本）的代码。
- 带有后缀 .f90、.f95、.f03 或 .f08 的文件被解释为自由格式。
- 带有大写后缀（例如 .F 或 .F90）的文件提示编译器对文件进行预处理。

本书中编写的所有代码都是自由格式的。自由格式简单地指的是在 Fortran 90 标准中引入的更自由的语法。由于我所知道的所有编译器都支持 .f90 作为自由格式代码的通用后缀（例如，Intel 编译器默认不支持 .f95、.f03 或 .f08），因此我们将在整本书中都使用这个扩展名。

A.2 设置 Fortran 编译器

有几个高质量的 Fortran 编译器可供选择。大多数由商业供应商如英特尔、Cray 等开发和维护。如果你能使用其中之一，那就太好了！在阅读本书时，可以随意使用它们。当然，对于任何特定于编译器的设置或使用说明，你需要参考你编译器的文档。

另外，作为 GNU 编译器集合（GCC）的一部分，还提供了一个免费、开源的 Fortran 编译器，即 GNU Fortran 编译器（gfortran，https://gcc.gnu.org/fortran）。在本书的示例和练习中，我们将使用 GNU Fortran 编译器。与其他编译器相比，gfortran 的优点包括：

- 免费下载、使用和修改
- 在大多数操作系统上易于安装
- 处于积极开发阶段
- 实现了大多数标准特性，包括一些来自最新的 Fortran 2018 标准

在编写本书时，出现了更多的开源编译器，并且仍在积极开发中。特别是，可以留意 LFotran（https://lfortran.org）和 Flang（https://github.com/flang-compiler/flang）。

Linux

在大多数 Linux 系统上，可以通过系统软件包管理器轻松安装 gfortran，而无需访问外部资源。在基于 DEB 的系统（如 Debian 或 Ubuntu）上，安装 gfortran 就像执行

```
apt install gfortran
```

一样简单。该命令会为您解决下载和安装步骤，一旦命令执行完成，gfortran 就可用了。在基于 RPM 的系统（如 Fedora 或 CentOS）上，您可以这样安装 gfortran：dnf install gcc-gfortran。或者，如果 gfortran 在系统的软件包管理器中不可用，您可以从 https://gcc.gnu.org/wiki/GFortranBinaries 下载二进制文件。

> 权限 您需要管理员（root）权限才能使用软件包管理器安装编译器。如果您知道自己在做什么，并且您的用户名已经在sudoers列表中，只需在我提供的Linux安装命令前加上sudo即可。

macOS

对于 macOS，我建议您使用 homebrew 包管理器（https://brew.sh）。一旦在您的系统上设置了 homebrew，安装 Fortran 编译器就像执行

```
brew install gcc
```

这个命令一样简单。该命令将安装基本的 GNU Compiler Collection 以及 Fortran 编译器。

Windows

在 Windows 中设置开发环境最简单的方法是通过 Windows 子系统来运行 Linux。这是一个在 Windows 10 操作系统中原生运行的 Ubuntu Linux 实例。如果您的 Windows 10 已经更新到最新版本，您可以从 Windows 应用商店获取 Ubuntu Linux 系统。一旦您安装好并运行起来，安装 Fortran 编译器就很容易了：

```
apt install gfortran
```

否则，有些人成功地在 Windows 上使用 Cygwin（https://www.cygwin.com）来开发 Fortran。

A.3 设置 MPI 库（消息传递接口）

在第 1 章中，我使用了一个处理器之间的数据复制示例来演示使用 MPI 进行并行编程。虽然在本书中我们将专注于使用 Coarray Fortran（CAF）进行并行算法，但我们仍然需要安装 MPI 库，因为它是 GNU 编译器在使用 coarrays 时的一个依赖项（请参阅“设置 OpenCoarrays”部分）。

我推荐使用 OpenMPI（https://www.open-mpi.org）或 MPICH（https://www.mpich.org）作为流行、高质量且易于使用的 MPI 实现。它们可以通过 Linux 包管理器安装。例如，如果您使用 Ubuntu 或其他基于 Debian 的发行版，您可以使用以下命令安装 OpenMPI：

```
apt install openmpi-bin libopenmpi-dev
```

在基于 RPM 的发行版（如 Fedora 或 CentOS）上，输入以下命令：

```
dnf install openmpi openmpi-devel
```

安装完成后，MPI 库会为编译器提供一个可执行包装器。如果安装正确，您可以通过在命令提示符下输入 mpif90 来验证：

```
mpif90
gfortran: fatal error: no input files compilation
terminated.
```

不要担心这个错误消息。这只是意味着 mpif90 在幕后正确调用了 gfortran，并且我们没有向它传递任何源文件。

A.4 设置 OpenCoarrays

OpenCoarrays（http://www.opencoarrays.org）提供了 GNU Fortran 编译器和底层并行实现（在我们的案例中是 MPI）之间的接口；你不需要知道更多。把它想象成 gfortran 的一个扩展，它将允许你使用 coarrays 构建和运行

并行程序。如果你已经可以访问带有 Intel 或 Cray 编译器套件的计算平台，你就不需要 OpenCoarrays，可以跳到下一节。

Linux

直接从其 GitHub 仓库获取 OpenCoarrays 发行版：

```
git clone --branch 2.9.0
https://github.com/sourceryinstitute/OpenCoarrays
```

通过源代码构建 OpenCoarrays 是最简单快速的方式。考虑到我们已经构建了 gfortran 和 OpenMPI，编译 OpenCoarrays 相对来说比较简单：

```
cd OpenCoarrays
mkdir build
cd build
FC=gfortran CC=gcc cmake ..
make
make install
```

你还需要 CMake（https://cmake.org）来构建 OpenCoarrays，并且需要 root 权限在你的系统上执行 make install。截至本文撰写时，OpenCoarrays 的最新版本是 2.9.0。然而，保持关注他们的发布页面（http://mng.bz/eQBG），如果有更新版本的话，请下载最新版本。

macOS

在 macOS 上使用 brew 安装 OpenCoarrays 非常简单：

```
brew install opencoarrays
```

使用 OpenCoarrays

OpenCoarrays 提供两个可执行文件：

- `caf` — 用于编译 Coarray Fortran 程序的包装脚本
- `cafrun` — 用于运行 Coarray Fortran 程序的包装脚本

编译 CAF 程序时，我们将使用 caf 作为我们的编译器的替代品；例如：

```
caf array_copy_caf.f90 -o array_copy_caf
```

要运行 CAF 程序，您将使用 cafrun 脚本调用它：

```
cafrun -n 2 array_copy_caf
```

此命令在两个并行进程上调用 array_copy_caf 程序。如果计算机上有两个物理处理器，则两者都将被使用。否则，cafrun 将在同一处理器上生成两个并行线程运行。这些细节不会影响程序的语义。

> 为什么我们需要 **OpenCoarrays**?
>
> Gfortran 完全支持 2008 年标准的 Fortran Coarray 语法。然而，单独使用 gfortran 还没有内置的机制来进行使用 coarrays 的并行计算。这意味着，使用普通的 gfortran，您可以编译 coarray 程序，但只能以单个图像（串行）模式运行它们。虽然这对于早期开发和测试可能很有用，但我们需要能够在多个图像上并行运行我们的程序。这就是 OpenCoarrays 的作用所在。
>
> 请注意，只有当您使用 gfortran 时才需要 OpenCoarrays。如果您使用的系统已经配置了 Intel 或 Cray 编译器套件，则可以使用构建 Coarray Fortran 代码。

A.5 构建 Docker 镜像

如果您熟悉 Docker，并希望跳过所有这些繁琐的设置，直接开始操作，请从 http://mng.bz/pBZR 下载 Dockerfile。要构建现代 Fortran 镜像，请键入：

```
docker build . -t modern-fortran:latest
```

此步骤需要一段时间，因为 Docker 将拉取基本操作系统镜像，并使用本书中的编译器、依赖项和 Fortran 代码设置镜像。一旦完成，如果构建成功，您将能够看到新的镜像；例如：

```
docker images
REPOSITORY           TAG       IMAGE ID       CREATED
SIZE
modern-fortran       latest    0e5c745c8928   6 minutes ago
546MB
```

要运行它，请键入：

```
docker run -it modern-fortran:latest /bin/bash
```

然后您就可以开始了！