写在最前面

(1) 此 FAQ（Frequently Asked Questions）针对 OpenFOAM（下面全部缩写为 OF）最开始的入门用户和学习者，仅供已经具备一定的 OF 使用经验的人参考。
(2) 此 FAQ 是由北航田超博士（QQ387210626-TCH 多物理场）整理，版权归原作者所有。

零基础相关

本节内容主要回答零基础学习者的遇到的常见问题，对于 OF 一无所知的初学者请看这一节，此节仅供高等学习者参考。

Question.1: 软件基础不是很好入门OF难不难？如何起步学习OF？

A：对于这个问题是排在第一位的常见问题。

传统的OF学习步骤是：

（1）Linux的初步使用和OF安装开始；

（2）通过安装和测试OF来获得对OF的基本认识；

（3）根据自己的问题来选择OF中最适合自己的求解器，求解器的具体功能参见后续的FAQ具体说明。【注：不要企图掌握OF中的所有内容，就像学习英语不要企图掌握牛津词典中所有单词一定，对于正常人尤其是入门的学习者，在精力上也是不现实的，初学OF比较忌讳企图掌握所有OF的solver和所有OF的功能，基于掌握OF所有的代码实现细节。试想OF是一个经验丰富的大师团队在近20年的学习和实践中积累起来的代码，不是对于OF生疏的吾辈能在短短的一到两年内能马上洞悉的。恳请抱有此想法的初学者尝试改变想法】；

（4）选择好最符合自己应用实际的求解器之后，认真做一到两个相关的算例，并根据自己的实际“需求”来定制修改求解器。【这里强调“需求”二字，需求一般具体为自己目前所做的算例工具链<从几何建模、网格划分、网格转换导入OF兼容格式、求解器参数设定计算、并行计算性能和计算时间、后处理和数据提取等等>是否符合自己要求】

（5）至此OF的学习已经完成了基础入门阶段内容了，下面的内容就是根据自己的需求定制自己的求解器并处理自己的问题了。■

Question.2: OF是什么？

A：OF是一套开源的代码，一般的开源代码都是写好了求解器，而后做好算例即可直接求解问题，遵循的是一条从问题到求解器再到结果的解决方法，但是OF是一套求解器的孵化器，而平时所接触的所有OF中的求解器，其实都是基于OF基本库通过编译连接得到的可以使用的求解器，遵循的是一调从问题到数学描述到求解器定制再到结算结果的解决方案。如果是初级用户可以通过直接使用OF的自带求解器来实现求解目的，而随着使用经验的增加，可以根据自己的问题，在已有的基础上修改求解器来适应自己的问题。

从技术层面上分析OF是什么需要参看后面的OpenFOAM算例相关部分。■

Question.3: OF只能在Linux下运行么？

A：OF最主流的使用方法是在Linux下运行，如果需要在Windows下运行，需要安装Linux系统的模拟器（MingW或者是Cygwin之类），在这些模拟器限定的范围内安装OF，在某些功能上可能难以与Linux下的使用效果相同。另外，在Windows系统下安装虚拟机系统，在虚拟机系统内运行OF系统，也是OF的典型应用方法之一。这也是一般的OF培训时最常用的方式，可以最大程度上省去OF的安装和配置过程中所面临的问题而直接针对OF使用本身开始学习。考虑到使用OF的用户都存在做大规模计算的需求，所以建议安装的Linux版本是x64架构下的，所下载的安装包一般内部携带有am64后缀（此处amd并非特指AMD公司处理器构架，而是泛指x64构架，intel的x64构架依然适用）或者x64后缀■

Question.4: OF是不是需要很多C++的基础知识啊？

A：对于OF的一般使用是不需要任何C++基础知识的，所涉及到的内容主要是运行OF过程中的Linux命令。如果想定制自己的OF求解器，那么必须要有一定的C++基础，在定制和修改求解器设置，定制全新的边界条件、湍流模型、限制器函数、传输模型以及一些设计到OF基本库使用的工具软件设计都需要对OF的C++代码有一定程度的了解。在修改定制自己的模型过程中，最主要用到的C++知识是构造函数、类的概念以及类成员函数的调用，对于初级求解器定制用户，封装性、继承性和多态性涉及的情况不多。初级求解器定制用户一般都是通过在现有的文件夹中选择相似的代码后批量修改关键字同时修改关键区域的函数代码来实现的。

注意：不建议为了看OF代码而先开始系统的学习C++，这样会浪费大量的时间而最后完全抓不住重点，OF和C++这两个技术的信息量都非常大。建议首先通过网络资源或者入门读物了解C++语言的基本语言点（主要是类的概念、类的定义、构造函数、初始化函数列表、析构函数、成员函数调用等、模板和模板类等），如果没有任何C++基础，建议入门书籍在保证信度的基础上越薄越好，一起迅速入门C++并在看OF代码的实践中慢慢学习。■

Question.5:OF开始起步学习要从头到尾看一遍User Guide（U-Guide）和Programming Guide（P-Guide）吗？这两个Guide有中文翻译版本吗？

A：不需要，入门阶段可以首先关注U-Guide最后的算例运行和使用部分，掌握OF的基本使用方法。U-Guide和P-Guide窃以为大部分内容也是给有一定基础的人使用的。上来就抱着两本书看到底效果有限。尚未发现质量较高的中文翻译版本，而且如果使用了中文翻译，可能在代码中呈现的注释和翻译过来的版本会出现出入，所以目前看英文版本是比较合适的版本。■

Question.6:能否简单描述一下OF的适用范围？

A：OF本质上是一套可以求解器偏微分方程（PDE）的程序库，很多程序库在底层功能上适用于从网格划分到PDE方程描述再到方程求解的全过程，在网格离散方面OF的finiteVolume库是其有限体积方法的核心库，所有的有限体积法的关键类和函数都来自于这个库的使用，所以OF在有限体积方法方面的适用范围很广，是一种通用的解决方法。但是，由于finiteVolume库中集成的网格离散方法是解耦的求解器（staggered solvers/pressure based solvers），对于Navier-Stokes方程的处理方法主要采用基于压力的修正方法（Simple类算法）速度和压力是通过不同的迭代不断修正得到的，所以OF最擅长于处理不可压缩的流动问题。而传统的可压缩流动求解器的主流处理方法是耦合求解器（coupled solvers/density based solvers），底层算法中速度、压力和温度参数是同时得到的，这种解法在OF内核中是无法简单的通过描述方程来实现的。当然，可以通过非OF风格的方法，直接写出可压缩耦合求解器的通量和迭代过程，弥补OF对于耦合求解器解法性能的不足。另外，OF还内置粒子模型、表面流模型、颗粒模型、DSMC模型等等求解器，对于非有限体积离散的PDE求解器，OF也是一个较为理想的选择。■

linux 初步相关

粗略估计 OF 群众讨论的问题，有一半以上的问题是 Linux 系统相关的问题，而主要的问题集中在权限错误，库缺失和对 Linux 中常见程序运行过程的误解上。

Question.7:Permission Denied是什么错误？

A：权限授权，linux系统中不论是否有管理员，都存在一个超级用户（super user或者叫root）权限，在向自己的“家目录”($HOME)以外的地方写数据、保存数据和覆盖数据的过程中，需要这个权限，和Windows中想修改C:盘Windows文件夹中的内容需要权限类似。在安装过程中，如果通过编译方式或者下载方式得到了可执行文件，而想拷贝到/usr/local、/usr/bin等等内核数据文件夹中，就需要这个权限，获得超级用户权限的方

法不同的系统中有所不同，在Ubunutu下面是在命令之前直接添加sudo命令。总之，如果遇到了Permission Denied错误提示，请首先想到需要给操作授予超级用户权限。■

Question.8: OF运行前有时候需要source <FOAM_INST_DIR>\etc\bashrc文件，这个文件是干什么用的？

A：本质上说，这个文件中写的数据都是与linux内核通信的脚本文件（shell脚本程序），这个文件中使用一系列的bash shell语法规则与linux内核通信，设定相关的环境变量，将必要的程序存储路径信息和环境变量信息提交给内核的查找路径下。这样在终端中运行程序的过程中，内核就会根据用户输入的命令查找相应的程序名称是否存储在查找路径下，如果查无此程序或文件夹就会报错。这个机制是操作系统中常用的机制。类似于Windows系统中对计算机\属性\高级\环境变量（用户变量和系统变量）等参数的操作。■

Question.9: gedit是我常用的编辑器，但是如果想修改一些没有写权限的文本文件的时候，gedit没法保存改如何处理啊？

A：使用gksudo gedit filename。此命令会询问授权超级用户权限，在得到权限后即可实现gedit对于权限不够时文本文件修改数据的写入。■

Question.10： OF到底用在哪个Linux发行版本中比较好啊？

A:技术上说OF可以应用于大多数的Linux发行版，比较常见的有Ubunut、OpenSUSE、Debian、Redhat、CentOS等等，笔者推荐使用CAELinux2011。CAELinux2011是基于Ubunut 10.04的专门面向CAE和科学计算单机使用的Linux发行版，里面预装了很多Linux和科学计算的软件，各种工具的备置都比较全面。同时，由于Ubunut10.04的版本是次新版本，所以很多更新版都新旧适中，比较适合初学者学习OF，降低OF软件配置时的出错概率。另外Ubunut的软件源较为丰富，所以在需要一些软件包的时候，可以直接使用新立得（Synaptic或者apt-get）安装。■

OpenFOAM 安装相关

在 OF 学习的典型建议步骤中，OF 自定义安装占有比较重要的地位，通过安装过程可以了解到 OF 程序模块配置的相关信息，同时也对 OF 基本系统结构有个整体上的把握。

Question.11： 对于标准版本的OF安装有没有较为详细的说明？

A：下面是笔者整理的一个较为详细的OF安装步骤说明，对于标准版本的OF安装而言，在CAELinux下，这个步骤是经过了很多次测试的：

（1） 首先到OF官方网站上下载OpenFOAM-2.1.1和ThirdParty-2.1.1的包（对于1.7.1以上的版本，下面的步骤都适用），只有这两个；

解压到$HOME/OpenFOAM如果没有此目录，最好建立一个，注意这个目录名称的大小写，OF安装的过程中，存在默认的路径，这个默认的路径都是基于$HOME/OpenFOAM的。

（2） 联网的Ubuntu下运行命令：

```
$ sudo apt-get install g++
$ sudo apt-get install zlib1g-dev
$ sudo apt-get install flex++
$ sudo apt-get install bison
$ sudo apt-get install binutils-dev
$ sudo apt-get install python
$ sudo apt-get install qt4-designer
$ sudo apt-get install cmake
$ sudo apt-get install libxt-dev
$ sudo apt-get install qt4-dev-tools
```

```
$ sudo apt-get install graphviz
```
（3）更新环境配置
【推荐步骤】安装kate（kate是一个文本编辑器，此步骤可选，只是笔者平时操作常用此编辑器）：只需要在控制台上输入(下面命令中kate是一个编辑器的名字，可以通过
```
$ sudo apt-get install kate
$ sudo apt-get install konsole
```
【必选步骤】更新环境变量。(注意：每次修改$HOME/OpenFOAM/OpenFOAM-2.1.1/etc/bashrc中的内容或者视图初始化OF时，都要做下面的操作！)
```
$ cd $HOME/OpenFOAM/OpenFOAM-2.1.1
$ source etc/bashrc
```
这样做的好处就是可以做多版本共存的OF，尤其是如果想让OF-ext和OF-Official共存的时候，想使用哪个版本，只需要source对应版本路径下的bashrc即可。
【推荐步骤】如果你想在debug模式下编译OF（debug模式下编译OF可以添加较多的调试信息，如果想跟踪OF内核代码，设定OF编译模式为debug是不可或缺的步骤），那么在$HOME/OpenFOAM/OpenFOAM-2.1.1/etc/bashrc中
设定
```
…
${WM_COMPILE_OPTION:=Debug}
export WM_COMPILE_OPTION=Debug
…
```
【推荐步骤】如果想设定多核编程选项，需要在etc/bashrc文件中添加下面语句(比如在 export WM_COMPILE_OPTION=Debug 下一行添加)，对多核编译环境变量导出：
```
…
export WM_NCOMPPROCS=2（注意没有空格！）
…
```
【推荐步骤】如果你要使用你系统的编译器（通常都需要改，ThirdParty里面没有gcc，所以学要使用系统自带的gcc），将/OpenFOAM/OpenFOAM-2.1.1/etc/bashrc中的
```
…
foamCompiler=system
…
```

（4）更新环境
通过刚才的所有修改后，需要再次更新bashrc文件
```
$ cd $HOME/OpenFOAM/OpenFOAM-2.1.1
$ source etc/bashrc
```
（5）编译第三方包
进入OpenFOAM/ThirdParty
```
$ cd $HOME/OpenFOAM/ThirdParty
$ ./Allwmake
```
（6）编译OpenFOAM
```
$ cd/OpenFOAM/OpenFOAM-1.7.1
$ cd $HOME/OpenFOAM/OpenFOAM-1.7.1
```
编译开始：
【不推荐步骤】$ ./Allwmake
或者

【推荐步骤】在编译过程中可能会出现问题，为了排错方便，可以将命令行出现的编译信息重新定向到文件。使用命令：
$ ./Allwmake &>make.log 2>&1

如果想重新编译，需要运行
$ ./wcleanAll

(7)编译paraview
$ cd $FOAM_INST_DIR/ThirdParty
$ ./Allclean
$ ./makeParaView
编译PVreader
$ cd $FOAM_UTILITIES/postProcessing/graphics/PV3FoamReader
$ ./Allwclean
$ ./Allwmake
(8)系统测试
在OpenFOAM-2.1.1安装目录的bin下运行
$ foamInstallationTest
看到核心模块和基本配置ok的提示后，表明配置成功。开始下一步的算例测试部分。

(9)算例测试，首先在$HOME/OpenFOAM下建立一个自己的工作目录
$ cp -r $FOAM_TUTORIALS $FOAM_RUN
$ cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavity
$ blockMesh
$ icoFoam
$ paraFoam（在此有些问题……）提示：OpenFOAM/OpenFOAM-2.1.1/bin/paraFoam: 142: paraview: not found
这表明paraView没有安装成功，运行$paraview提示没有安装，那么使用sudo apt-get install paraview来解决此问题。
OpenFOAM的标准安装过程中包含有paraview的安装过程，如果需要使用paraview而且同时可能会使用OpenFOAM的话，那么最好直接安装OpenFOAM即可。

Question.12: OF安装需要花费很多时间吗？我编译一次用12个小时正常吗？为什么有人编译一次只用3个小时？

A：OF的编译过程是挺消耗时间的，在各种软件齐备的情况下，OF正常编译一次的时间大概是3到6个小时左右，根据不同的电脑配置性能会有明显差异。为了提高编译速度，建议编译过程中导出打开并行编译开关。即在$HOME/OpenFOAM/OpenFOAM-2.1.1/etc/bashrc中设定
…
export WM_NCOMPPROCS=2
…
此时是双核编译，双核一起生成目标文件，理论上可以线性提升编译速度。如果是12和的电脑，可以直接配置为export WM_NCOMPPROCS=12，编译加速会有非常明显的提升。

OpenFOAM 算例相关

Question.13: 使用OF用什么网格划分软件？

A：向OF导入兼容格式的网格途径很多。主要分为三类，下面做一简略说明。
第一种是使用OF自带的blockMesh和SnappyHexMesh（简称SHM）工具来生成；
这种方法得到的OF网格是直接可以使用的，blockMesh的使用方法也在U-Guide中做了详细的说明，此处不再赘述；SHM的使用中参数设定较复杂，建议从motorBike算例中的snappyHexMeshDict修改为和自己的算例兼容的配置文件。
第二类是使用现有其他格式的网格通过OF自带的网格转换工具向OF转化；
ICEM划分的网格主要就是通过转化为Fluent兼容的网格，然后使用OF中的fluent网格转化工具来实现的。
第三类是使用外部商业软件或独立代码直接生成OF兼容网格
可以生成OF兼容网格的商业软件主要有：ANSA、Gridpro（在网上可以下载得到Gridpro2Foam转换工具）、Gridgen/Pointwise、enGrid、Cubit、StarCD/StarCCM+。
这三种网格生成途径没有特别的侧重，而且都挺常用，学习者可以根据自己的具体需求来选用工具。

OpenFOAM 的算例目录和文件结构相关

OpenFOAM 内核源代码初步问题相关

Question.14: 在OF下如何进行调试？

A：在OF下的调试是比较困难的，尤其是跟踪内核级别的调试，有些代码段可能并没有调试信息，所以比较难定位错误的位置。如果非要调试的话，可以使用gdb工具来跟踪代码查看变量和寄存器内信息和堆栈数据。另外，调试的本质上是一个排错的过程，广义上来说，排错的过程都是调试过程，一些常用的方法：

（1） 使用info函数直接将数据输出；
（2） 通过减小时间步长通过ParaView来精细的查看流场中出错位置，很多的错误是扩散而来的，比如说从边界条件扩散而来的错误；
（3） 遵循基本的代码开发过程，通过标准算例和知道信度的算例来校核自己的代码，比如说知道cavity算例各个点数据的分布，更理想的情况是知道解析解，然后将自己的数值解和解析解进行比较；

定位错误的来源对于数据量大的CFD计算代码而言的确充满了困难。经常出现的错误就是segmentation fault和floating point exception，这两种错误来自于数据的非法访问（比如说没有初始化的数据，没有new的数据你进行了访问）和零做了除数或者负数开了平方。
总之，调试就是定位错误然后排错，和经验有很大关系。最有效的方法就是使用上面建议的第三条。先从简单的算例慢慢开始，只有自己确定算出来的结果是什么，才能知道自己测试代码的方向和问题可能存在的位置。定位了问题，排错都是水套渠成的事情。
■