OpenFOAM 探险队(原名 OpenFOAM たんけんたい)

原作：Yuu Kasuga (penguinitis2002@yahoo.co.jp)
翻译：HoTsAUce (houkensjtu@gmail.com qq:305649246)

作者序：
为了潜入 OpenFOAM 源代码茂密的丛林深处，我们组织了这支探险队。以下是活动报告。

第三期 Time 类

目的：
让我们试用一下 Time 这个类吧。

使用版本：
OpenFOAM 1.6（译者注：译者用 2.1.x 版也大都能够编译通过并成功运行）

程序说明：
和上回一样，首先我们需要一个容纳源代码的目录。然后将压缩包中的内容在其中解压后，会发现如下文件：

- Make/files
- Make/options
- system/controlDict
- Time.C

Make/files

```
Time.C

EXE = Time
```

files 中指定了本次将生成的可执行文件的名称：Time。

options 文件基本和上次一样。

Make/options

```
EXE_INC = -I$(LIB_SRC)/finiteVolume/lnInclude
EXE_LIBS = -lfiniteVolume
```

接下来就是这次的重头戏了，OpenFOAM 中作为惯例，一般会通过读取“XXXDict”也就是什么什么字典的形式来进行设置参数。这次我们用到的 Time 类，需要（必须）一个 controlDict 来读取相关参数，因此我们得准备一个 controlDict。具体做法就是，建立一个 system/目录，然后，从 tutorial 文件夹里随便拷贝一个过来就可以啦，比如说，tutorials/incompressible/icoFoam 的 cavity 下的 system 目录内就有。这次我们主要设定并试用的是如下这些参数：

- startTime：计算开始时间
- endTime：计算结束时间
- deltaT：时间步长
- writeInterval：写数据的时间间隔

icoFoam 等一些一般的求解器的 tutorial 中，都是从 startTime 开始，共进行(endTime-startTime)/deltaT 次的计算，在这过程之中，每隔 writeInterval 的时间，就写出一次数据。（译者注：但是也有例外比如，如果 deltaT 是 1 秒，而 writeInterval 是 0.5 秒，那么因为每隔 0.5 秒的数据也都需要输出，所以 openFoam 一般都会自动每隔 0.5 秒进行一次计算哦）

Time.C 是本次的，为了理解 Time 类所编写的程序。（头尾略）

```
...
    Time runTime(Time::controlDictName,
            args.rootPath(), args.caseName());

    Info<<"controlDict name: "<<runTime.controlDictName<<endl;
    Info << "root path: " << runTime.rootPath() << endl;
    Info << "case name: " << runTime.caseName() << endl;
    Info << "path: " << runTime.path() << endl;
    Info << "time path: " << runTime.timePath() << endl;
    Info << "format: " << runTime.writeFormat() << endl;
    Info << "version: " << runTime.writeVersion() << endl;
    Info<<"compression: "<<runTime.writeCompression()<<endl;

    Info << "start time: " << runTime.startTime() << endl;
    Info << "end time: " << runTime.endTime() << endl;
    Info << "deltaT: " << runTime.deltaT() << endl;

    runTime.writeNow();

    while(runTime.loop()){
```

```
            Info << "Time: " << runTime.timeName() << endl;
            runTime.write();
    }

    runTime.writeAndEnd();

    Info << "execution time: " << runTime.elapsedCpuTime() << " s"
<< endl;
    Info << "clock time: " << runTime.elapsedClockTime() << " s" <<
endl;
    ...
```

Time 这个类负责管理计算过程中所有管理任务。不光是时间相关，包括程序的路径等情报，另外何时进行数据写出也都由它来担当。程序本身相关情报是通过读取 argList 类的对象（上回介绍的）以及 controlDict 来完成。具体的情况请自己探索 Time 的源代码。一般在这个目录下可以找到 src/OpenFOAM/db/Time/Time.H（译者注：探索代码的时候不可能把每个代码的位置都记住吧？试试 find $FOAM_SRC –name Time.H）

Time.H 中的 public 部分是用户可以呼叫的函数列表，具体查阅一下就可以知道，用这个类可以干嘛了，比如：

```
public:
    ...
    //- Return current time name
    virtual word timeName() const;
    ...
```

看到这一行，就知道，用 timeName()这个函数可以返回出当前时间的名称。

“while(runtime.loop()){...}”是循环计算的外部结构，具体循环多少次就是通过读取 controlDict 中的相关信息来决定的。write()就是写出数据，具体什么时间点写出数据，那当然是根据 controlDict 的 writeInterval 来决定啦。

编译与执行：

在代码目录下（不要进到 Make 目录里），执行 wmake 即可编译。如果没有问题的话，当前目录下会产生一个叫做 Time 的文件，执行试试看吧~

```
$ wmake
$ ./Time
```

彩蛋：

再捣腾一下原来写好的代码吧~代码包是 03-time2.tar.gz，解压以后可以发现，代码修改的部

分如下：

Time.C（头尾省略）

```
...
Info << "start time: " << runTime.startTime() << endl;
Info << "end time: " << runTime.endTime() << endl;
Info << "deltaT: " << runTime.deltaT() << endl;

Info << "*set end time = 10" << endl;
runTime.setEndTime(10);

Info << "*set DeltaT = 2" << endl;
runTime.setDeltaT(2);

Info << "start time: " << runTime.startTime() << endl;
Info << "end time: " << runTime.endTime() << endl;
Info << "deltaT: " << runTime.deltaT() << endl;
...
```

到底是在修改神马呢？其实就是，在程序内部修改了 endTime 和 deltaT 的值。想要修改某些计算参数的时候，除了更改 controlDict 并再次读取，如果看一看相关类的成员函数的话，说不定会发现已经有现成的直接可以用来修改数据的函数哦。

彩蛋 2：

Time 类的成员函数 timeName()的返回型是 Foam::word。这说白了是一个文字串，那如果我想获得一个当前时间的数值，而不是字符呢？以下的方法可行（也许有更好的，我并不确定）

```
double currentTime = atof(runTime.timeName().c_str());
```

具体分析：

1. Time 类的 timeName()函数将现在的时间名称以 Foam::word 型返回
2. Foam::word 继承的是 Foam::string
3. Foam::string 继承的是 std:string
4. Std::string 类的成员函数 c_str()将返回一个 C 风格的字符串
5. C 的库函数 atof 可以将字符串转换成浮点数
6. 这样 currentTime 就得到当前时间的数值了

（译者注：原作者好像是一个长期使用 C 语言的用户，因此这里的转换方法也是 C 风格的方法，有没有办法用 c++的方法来完成这件事呢？）