OpenFOAM 中的字典

2019-04-26 | code analysis | 字数总计: 2k | 阅读时长: 10 分钟

OpenFOAM 中的 dictionary 和 IOdictionary

常见用法

创建 IOdictionary

```c++
IOdictionary transportProperties//字典的变量名
(
    IOobject
    (
        "transportProperties",//字典名字
        runTime.constant(),   //字典位于constant下
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
```

创建字典并不是创建文件!

如果想自定义路径呢?

可以把 `runTime.constant()` 改成 `"./Properties"` 。

这样的话，我们就需要在当前算例目录下新建一个文件夹 `Properties` ，文件夹里需要有一个文件 `transportProperties` 。

该文件还必须包含文件头（header）：

```c++
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
```

里边的信息可以随意，比如 `location` 仍然写 `constant` 也没关系。

使用已有 IOdictionary

```c++
const dictionary& subModelDict = parcels.subModelProperties();//这里获取已有的字典
vector position =
subModelDict.subDict("injectionModels").subDict("model1").lookup("position");
//subDict 是子字典
```

目录

这里的 parcels.subModelProperties() 是 SprayCloudProperties 中的 subModels。

```c++
const wordList fuelList(thermo.subDict("liquids").keys());
```

这里的 thermo 是 thermophysicalProperties。

通过字典读取变量

```c++
word fuel(dict.lookup("fuel"));
dimensionedScalar DT(dict.lookup("DT"));
scalarList Z_param(dict.lookup("Z_param"));
scalar Sct = readScalar(dict.lookup("Sct"));
label int_a=readLabel(dict.lookup("int_a"));
Switch flagg(dict.lookup("flagg"));
```

另外还可以给定默认值，即 lookupOrdefault 函数：

```c++
word fuel2(dict.lookupOrDefault("fuel2",fuel));
dimensionedScalar DT2(dict.lookupOrDefault("DT2",DT));
scalarList Z_param2(dict.lookupOrDefault("Z_param2",Z_param));
scalar Sct2 = dict.lookupOrDefault("Sct2", 9.);
label int_a2(dict.lookupOrDefault("int_a2", 25));
Switch flagg2(dict.lookupOrDefault("flagg2",false));

//或者：
word fuel3(dict.lookupOrDefault<word>("fuel3",fuel));
dimensionedScalar DT3(dict.lookupOrDefault<dimensionedScalar>("DT3",DT));
scalarList Z_param3(dict.lookupOrDefault<scalarList>("Z_param3",Z_param));
scalar Sct3 = dict.lookupOrDefault<scalar>("Sct3", 9.);
Switch flagg3(dict.lookupOrDefault<Switch>("flagg3",false));
label int_a3(dict.lookupOrDefault<label>("int_a3", 35));
```

算例文件：

```c++
fuel ZY.3;// word 中允许有 点 ?
DT DT [0 2 -1 0 0 0 0] 1e-05;
//Z_param (0 0.2 1);  //OK
//Z_param 3(0 0.2 1); //OK too
Z_param 101{0};       //OK too
Sct 10.2;
flagg true; //expected 'true/false', 'on/off'

Sct2 10.8;
```

如果重复给定，那么最后边的那个会起作用。

给定一个全是 0 的多维数组：

```c++
zeroList
21
{
```

目录

你已经读了 0 %


```
4       308
5       {
6           101{0}
7       }
8   }
9   ;
```

看一下这两个函数的代码，发现 lookup 返回的是 ITstream，而 lookupOrDefault 返回的是它的模板类型。

疑问：

- 为什么需要 readScalar 才能使用 lookup 读取标量？可能是因为 ITstream 无法隐式的转换成 scalar。
- 为什么 lookupOrDefault 不给定模板也可以呢？因为它的第二个参数的类型就是模板的类型啊。

c++

```
1   ITstream& lookup(const word&,bool recursive=false,bool patternMatch=true) const;
2
3   Foam::ITstream& Foam::dictionary::lookup
4   (
5       const word& keyword,
6       bool recursive,
7       bool patternMatch
8   ) const
9   {
10      return lookupEntry(keyword, recursive, patternMatch).stream();
11  }
12
13  const Foam::entry& Foam::dictionary::lookupEntry
14  (
15      const word& keyword,
16      bool recursive,
17      bool patternMatch
18  ) const
19  {
20      const entry* entryPtr = lookupEntryPtr(keyword, recursive, patternMatch);
21
22      if (entryPtr == nullptr)
23      {
24          FatalIOErrorInFunction
25          (
26              *this
27          )   << "keyword " << keyword << " is undefined in dictionary "
28              << name()
29              << exit(FatalIOError);
30      }
31
32      return *entryPtr;
33  }
34
35
36  template<class T>T lookupOrDefault(const word&,const T&,bool recursive=false,bool
patternMatch=true) const;
37
38  template<class T>
39  T Foam::dictionary::lookupOrDefault
40  (
41      const word& keyword,
42      const T& deflt,
43      bool recursive,
```

目录

你已经读了 0 %


```
    bool patternMatch
) const
{
    const entry* entryPtr = lookupEntryPtr(keyword, recursive, patternMatch);

    if (entryPtr)
    {
        return pTraits<T>(entryPtr->stream());
    }
    else
    {
        if (writeOptionalEntries)
        {
            IOInfoInFunction(*this)
                << "Optional entry '" << keyword << "' is not present,"
                << " returning the default value '" << deflt << "'"
                << endl;
        }

        return deflt;
    }
}


const Foam::entry* Foam::dictionary::lookupEntryPtr
(
    const word& keyword,
    bool recursive,
    bool patternMatch
) const
{
    HashTable<entry*>::const_iterator iter = hashedEntries_.find(keyword);

    if (iter == hashedEntries_.end())
    {
        if (patternMatch && patternEntries_.size())
        {
            DLList<entry*>::const_iterator wcLink =
                patternEntries_.begin();
            DLList<autoPtr<regExp>>::const_iterator reLink =
                patternRegexps_.begin();

            // Find in patterns using regular expressions only
            if (findInPatterns(patternMatch, keyword, wcLink, reLink))
            {
                return wcLink();
            }
        }

        if (recursive && &parent_ != &dictionary::null)
        {
            return parent_.lookupEntryPtr(keyword, recursive, patternMatch);
        }
        else
        {
            return nullptr;
        }
    }

    return iter();
}
```

无网格创建 IOdictionary

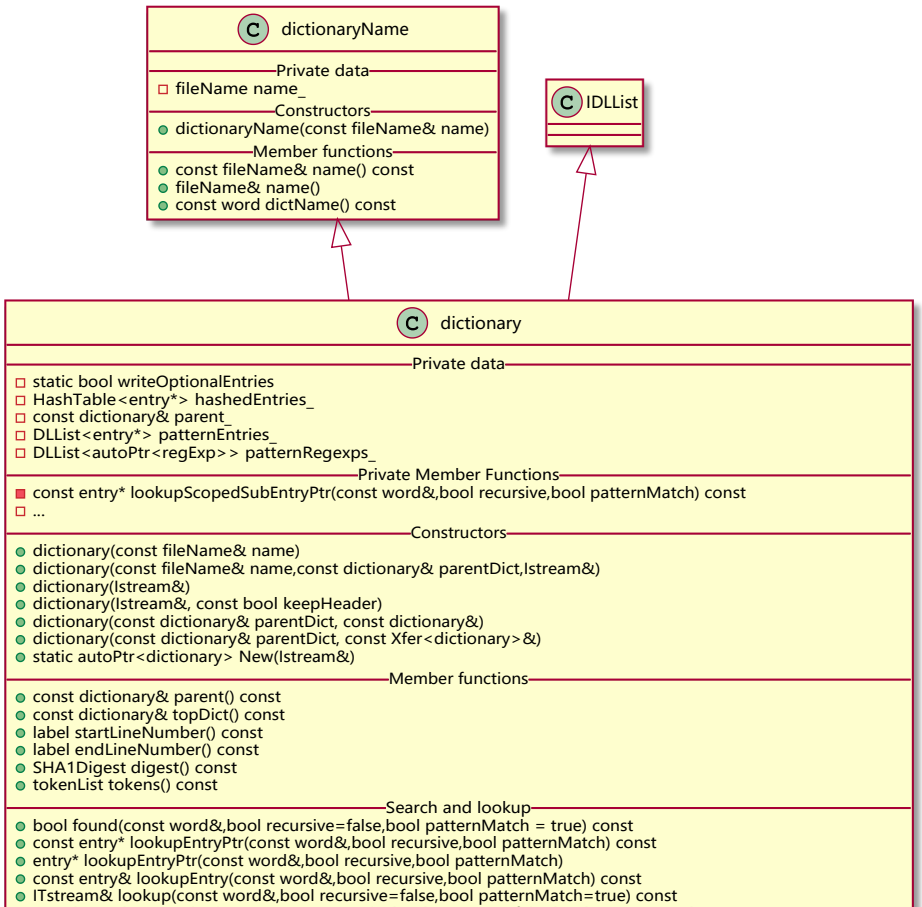
目录

这段代码来自与Zmeng的私下交流。

plain

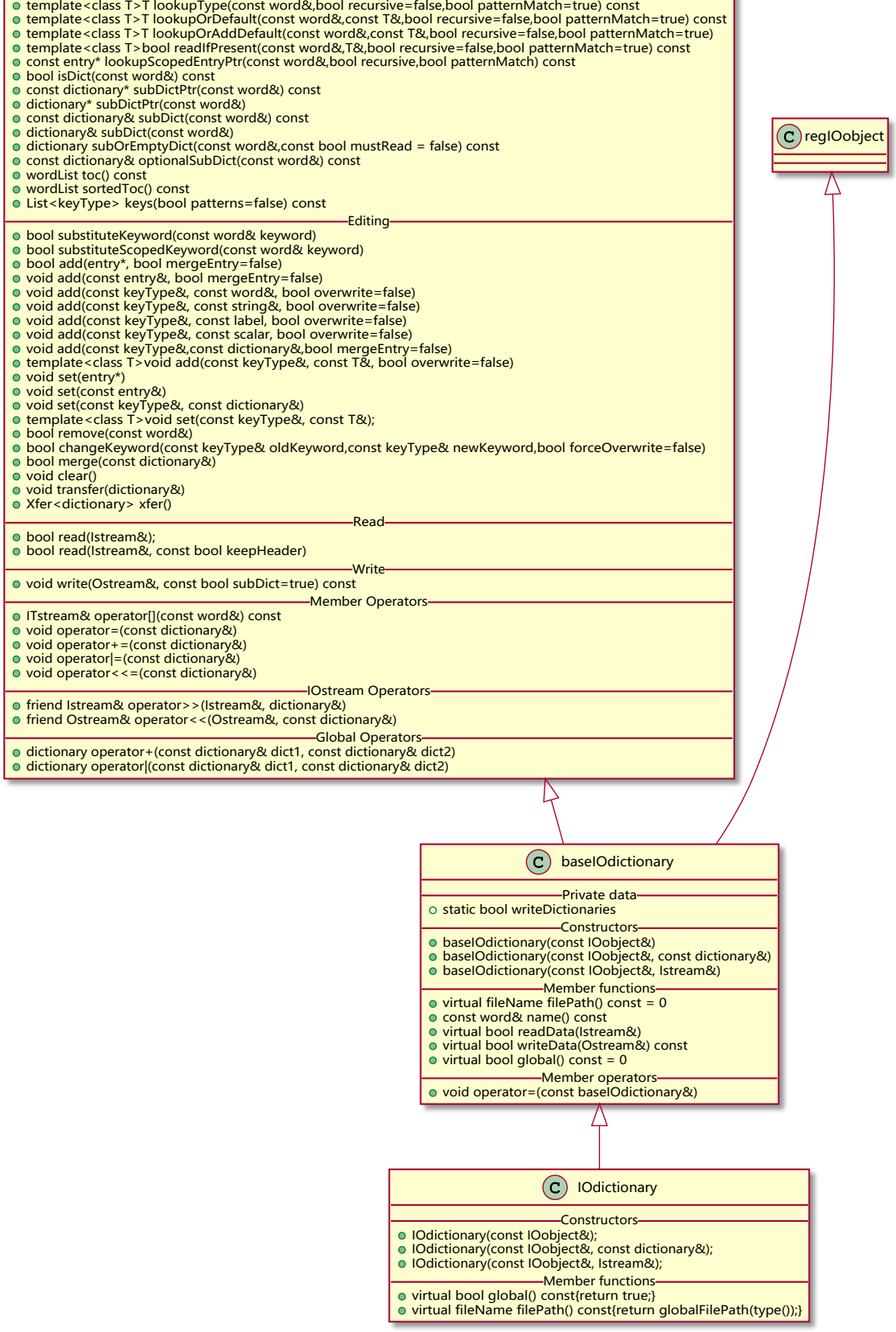
```
Info<< "Reading thommieDict\n" << endl;

const fileName caseDirOrig = getEnv("FOAM_CASE");
const fileName rootDirSource = caseDirOrig.path();
const fileName caseDirSource = caseDirOrig.name();

Info<< "Source: " << rootDirSource << " " << caseDirSource << endl;

Time runTimeSource
(
    Time::controlDictName,
    rootDirSource,
    caseDirSource
);

IOdictionary thommieDict
(
    IOobject
    (
        "thommieDict",
        runTimeSource.system(),
        runTimeSource,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);

const dictionary geometrys(thommieDict.subDict("geometrys"));
```

代码解析



目录

你已经读了 0 %



一些函数探秘

算例里的文件片段：

c++

```
RanzMarshallCoeffs
{
    BirdCorrection  true;
}
```

目录

代码：

```c++
    const dictionary& subModelDict =
parcels.subModelProperties().subDict("RanzMarshallCoeffs");
    Info<<"RanzMarshallCoeffs.toc()==="<<subModelDict.toc()<<endl;
    Info<<"RanzMarshallCoeffs.keys()==="<<subModelDict.keys()<<endl;
```

输出：

```c++
    RanzMarshallCoeffs.toc()===1(BirdCorrection)
    RanzMarshallCoeffs.keys()===1(BirdCorrection)
```

算例里的文件片段：

```c++
    liquids
    {
       C7H16
        {
            defaultCoeffs   yes;
        }
        IC8H18
        {
            defaultCoeffs   yes;
        }
    }
```

代码：

```c++
    const wordList fuelList(thermo.subDict("liquids").keys());
    const wordList fuelCoeff1(thermo.subDict("liquids").subDict("C7H16").keys());
    Info<<"fuelList==="<<fuelList<<endl;
    Info<<"fuelCoeff1==="<<fuelCoeff1<<endl;
```

输出：

```c++
    fuelList===
    2
    (
    C7H16
    IC8H18
    )

    fuelCoeff1===1(defaultCoeffs)
```

算例里的文件片段：

```c++
    Cd   constant 0.88;
```

目录

其实这里的 Cd 是读取给一个 TimeFunction1<scalar> 的类型。

这里我们强行读取后边的标量：

```c++
    scalar Cd_ = 0.;
    token firstToken(modelDict.lookup("Cd")[1]);
    //这里 lookup 得到一个list，我们取[1]，即后边的 0.88
    if (firstToken.isScalar())
    {
        Cd_ = firstToken.scalarToken();
    }
```

其实"更"正确的用法是：

```c++
    TimeFunction1<scalar> Cd2_(mesh.time(),"Cd");
    Cd2_.reset(modelDict);
    Info<<"Cd2_==="<<Cd2_.value(0.)<<endl;
    scalar Cd_ = Cd2_.value(0.);
```

算例里的文件片段：

```c++
    37 liquids
    38 {
    39           C7H16
    40     {
    41         defaultCoeffs   yes;
    42     }
    43     IC8H18
    44     {
    45         defaultCoeffs   yes;
    46     }
    47 }
```

代码：

```c++
    Info<<"thermo.subDict(liquids).subDict(C7H16).parent()==="
<<thermo.subDict("liquids").subDict("C7H16").parent()<<endl;
    Info<<"thermo.subDict(liquids).topDict()==="<<thermo.subDict("liquids").topDict()<<en
    Info<<"thermo.subDict(liquids).startLineNumber()==="<<thermo.subDict("liquids").start
    Info<<"thermo.subDict(liquids).endLineNumber()==="<<thermo.subDict("liquids").endLine
    Info<<"thermo.subDict(liquids).tokens()==="<<thermo.subDict("liquids").tokens()<<endl
    Info<<"thermo.subDict(liquids).toc()==="<<thermo.subDict("liquids").toc()<<endl;
```

输出：

parent 返回的是上一级字典。

topDict 返回的是最顶层的字典。

```c++
    thermo.subDict(liquids).startLineNumber()===41
    thermo.subDict(liquids).endLineNumber()===45
    thermo.subDict(liquids).tokens()===
    12
```

目录


```
(
C7H16
{
defaultCoeffs
yes
;
}
IC8H18
{
defaultCoeffs
yes
;
}
)

thermo.subDict(liquids).toc()===
2
(
C7H16
IC8H18
)
```

tokens 就是这个字典中的所有元素，这里有 12 个元素，包括字符串和标点符号。

而 toc 是 keys 一样的？他们的定义如下：

c++

```c++
Foam::wordList Foam::dictionary::toc() const
{
    wordList keys(size());

    label nKeys = 0;
    forAllConstIter(IDLList<entry>, *this, iter)
    {
        keys[nKeys++] = iter().keyword();
    }

    return keys;
}

Foam::List<Foam::keyType> Foam::dictionary::keys(bool patterns=false) const
{
    List<keyType> keys(size());

    label nKeys = 0;
    forAllConstIter(IDLList<entry>, *this, iter)
    {
        if (iter().keyword().isPattern() ? patterns : !patterns)
        {
            keys[nKeys++] = iter().keyword();
        }
    }
    keys.setSize(nKeys);

    return keys;
}
```

踩过的坑

我想读取一个文件，但是并行时只想让 master 读取。

目录

初次尝试

首先构造一个 `IOdictionary`，NO_READ。

plain

```
IOdictionary Dict
(
  IOobject
  (
     "dummyName",//tableName
     path,
     mesh,
     IOobject::NO_READ,
     IOobject::NO_WRITE
  )
);
```

然后在 master 读取

plain

```
if (onlyMasterRead&&Pstream::parRun())
{
    if (Pstream::master())
    {
        Dict.readOpt() = IOobject::MUST_READ;
        Dict.readData(this->readStream(typeName));
        Dict.close();
    }
}
else
{
    Dict.readOpt() = IOobject::MUST_READ;
    Dict.readData(this->readStream(typeName));
    Dict.close();
}
```

这段代码在本地测试时没问题，但是提交到超算之后就报错了：

plain

```
[1] --> FOAM FATAL IO ERROR:
[1] error in IOstream "IOstream" for operation operator>>(Istream&, List<T>&) :
reading first token
[1]
[1] file: IOstream at line 0.
[1]
[1]     From function void Foam::IOstream::fatalCheck(const char*) const
[1]     in file db/IOstreams/IOstreams/IOstream.C at line 109.
[1] FOAM parallel run exiting
```

注意到这里master没有报这个错，其它核都报了。

因此猜测，IOdictionary必须每个核保持一直，不能有差异！

再次尝试

首先构造一个 `IOdictionary`，读取一个特别小的 dummyName。

目录

plain

```
IOdictionary Dict
(
  IOobject
  (
     "dummyName",//tableName
     path,
     mesh,
     IOobject::MUST_READ,
     IOobject::NO_WRITE
  )
);
```

然后在 master 读取真正的文件

plain

```
if (onlyMasterRead&&Pstream::parRun())
{
    if (Pstream::master())
    {
        Dict.rename(tableName);
        Dict.readData(this->readStream(typeName));
        Dict.close();
    }
}
else
{
    Dict.rename(tableName);
    Dict.readData(this->readStream(typeName));
    Dict.close();
}
```

还是和上边一样的问题!

解决方案

不用 `IOdictionary`，而是用 `dictionary`

plain

```
if (onlyMasterRead&&Pstream::parRun())
{
    if (Pstream::master())
    {
        tableValues_ = dictionary(IFstream(tablePath+"/"+tableName)
()).lookupOrDefault<List<scalar>>(tableName, defaultList);
    }
    Pstream::scatter(tableValues_);
}
else
{
    tableValues_ = dictionary(IFstream(tablePath+"/"+tableName)
()).lookupOrDefault<List<scalar>>(tableName, defaultList);
}
```

目录

你已经读了 0 %

dictionary IOdictionary

您的肯定会给我更大动力~

OpenFOAM Learning Resources

OpenFOAM 中的喷雾速度计算

0 条评论 未登录用户

说点什么

支持 Markdown 语法 使用 GitHub 登录 预览

来做第一个留言的人吧!