

目录

你已经读了 0 %

1. 常见用法
2. 代码解析
3. 踩过的坑

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++

```
1  IOdictionary transportProperties//字典的变量名
2  (
3      IOobject
4      (
5          "transportProperties",//字典名字
6          runTime.constant(),    //字典位于constant下
7          mesh,
8          IOobject::MUST_READ_IF_MODIFIED,
9          IOobject::NO_WRITE
10     )
11 );
```

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++

```
1  FoamFile
2  {
3      version      2.0;
4      format        ascii;
5      class         dictionary;
6      location      "constant";
7      object        thermophysicalProperties;
8  }
```

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++

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

目录

你已经读了 0 %

1. 常见用法
2. 代码解析
3. 踩过的坑

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

C++

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

这里的 `thermo` 是 `thermophysicalProperties`。

通过字典读取变量

C++

```
1 word fuel(dict.lookup("fuel"));
2 dimensionedScalar DT(dict.lookup("DT"));
3 scalarList Z_param(dict.lookup("Z_param"));
4 scalar Sct = readScalar(dict.lookup("Sct"));
5 label int_a=readLabel(dict.lookup("int_a"));
6 Switch flagg(dict.lookup("flagg"));
```

另外还可以给定默认值，即 `lookupOrDefault` 函数：

C++

```
1 word fuel2(dict.lookupOrDefault("fuel2",fuel));
2 dimensionedScalar DT2(dict.lookupOrDefault("DT2",DT));
3 scalarList Z_param2(dict.lookupOrDefault("Z_param2",Z_param));
4 scalar Sct2 = dict.lookupOrDefault("Sct2", 9.);
5 label int_a2(dict.lookupOrDefault("int_a2", 25));
6 Switch flagg2(dict.lookupOrDefault("flagg2",false));
7
8 //或者:
9 word fuel3(dict.lookupOrDefault<word>("fuel3",fuel));
10 dimensionedScalar DT3(dict.lookupOrDefault<dimensionedScalar>("DT3",DT));
11 scalarList Z_param3(dict.lookupOrDefault<scalarList>("Z_param3",Z_param));
12 scalar Sct3 = dict.lookupOrDefault<scalar>("Sct3", 9.);
13 Switch flagg3(dict.lookupOrDefault<Switch>("flagg3",false));
14 label int_a3(dict.lookupOrDefault<label>("int_a3", 35));
```

算例文件：

C++

```
1 fuel ZY.3; // word 中允许有 点 ?
2 DT DT [0 2 -1 0 0 0 0] 1e-05;
3 //Z_param (0 0.2 1); //OK
4 //Z_param 3(0 0.2 1); //OK too
5 Z_param 101{0}; //OK too
6 Sct 10.2;
7 flagg true; //expected 'true/false', 'on/off'
8
9 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++

```
1 zeroList
2 21
3 {
```

目录

你已经读了 0 %

1. 常见用法
2. 代码解析
3. 踩过的坑

```

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 %

1. 常见用法
2. 代码解析
3. 踩过的坑

```

44     bool patternMatch
45 ) const
46 {
47     const entry* entryPtr = lookupEntryPtr(keyword, recursive, patternMatch);
48
49     if (entryPtr)
50     {
51         return pTraits<T>(entryPtr->stream());
52     }
53     else
54     {
55         if (writeOptionalEntries)
56         {
57             IOInfoInFunction(*this)
58                 << "Optional entry '" << keyword << "' is not present,"
59                 << " returning the default value '" << deflt << "'"
60                 << endl;
61         }
62
63         return deflt;
64     }
65 }
66
67
68 const Foam::entry* Foam::dictionary::lookupEntryPtr
69 (
70     const word& keyword,
71     bool recursive,
72     bool patternMatch
73 ) const
74 {
75     HashTable<entry*>::const_iterator iter = hashedEntries_.find(keyword);
76
77     if (iter == hashedEntries_.end())
78     {
79         if (patternMatch && patternEntries_.size())
80         {
81             DLList<entry*>::const_iterator wLink =
82                 patternEntries_.begin();
83             DLList<autoPtr<regExp>>::const_iterator reLink =
84                 patternRegexps_.begin();
85
86             // Find in patterns using regular expressions only
87             if (findInPatterns(patternMatch, keyword, wLink, reLink))
88             {
89                 return wLink();
90             }
91         }
92
93         if (recursive && &parent_ != &dictionary::null)
94         {
95             return parent_.lookupEntryPtr(keyword, recursive, patternMatch);
96         }
97         else
98         {
99             return nullptr;
100         }
101     }
102
103     return iter();
104 }

```



目录

你已经读了 0 %

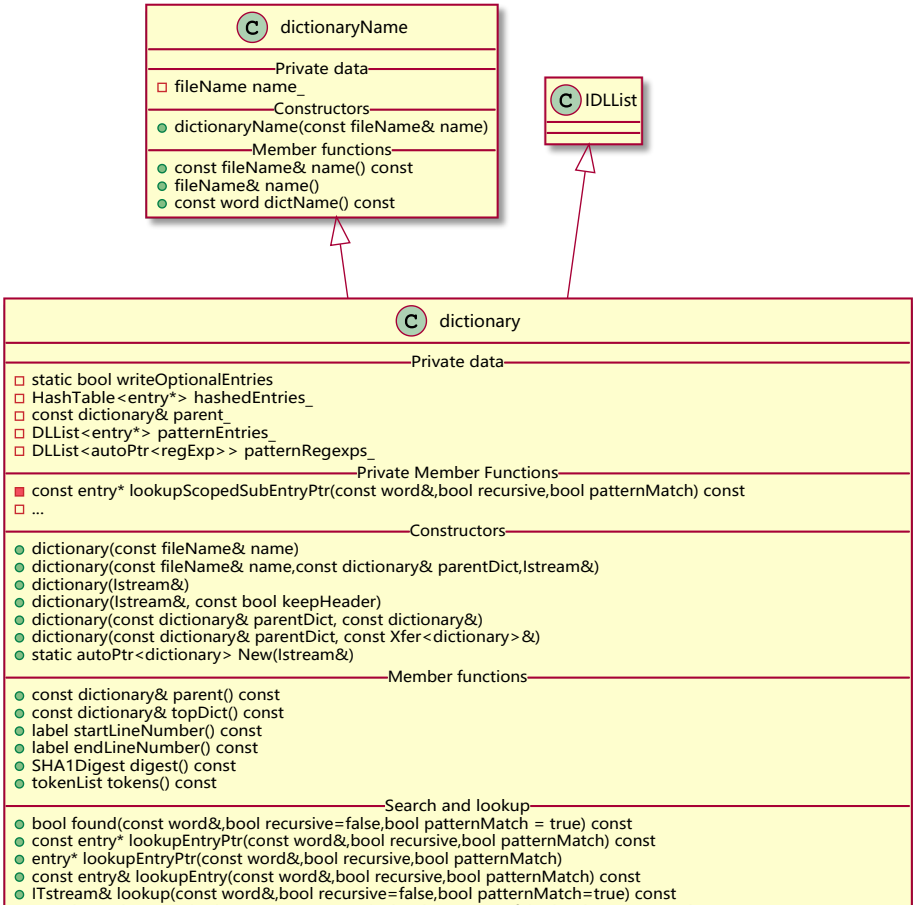
- 1. 常见用法
- 2. 代码解析
- 3. 踩过的坑



plain

```
1 Info<< "Reading thommieDict\n" << endl;
2
3 const fileName caseDirOrig = getEnv("FOAM_CASE");
4 const fileName rootDirSource = caseDirOrig.path();
5 const fileName caseDirSource = caseDirOrig.name();
6
7 Info<< "Source: " << rootDirSource << " " << caseDirSource << endl;
8
9 Time runTimeSource
10 (
11     Time::controlDictName,
12     rootDirSource,
13     caseDirSource
14 );
15
16 IOdictionary thommieDict
17 (
18     IOobject
19     (
20         "thommieDict",
21         runTimeSource.system(),
22         runTimeSource,
23         IOobject::MUST_READ_IF_MODIFIED,
24         IOobject::NO_WRITE
25     )
26 );
27
28 const dictionary geometrys(thommieDict.subDict("geometrys"));
```

代码解析





目录

你已经读了 0 %

- 1. 常见用法
- 2. 代码解析
- 3. 踩过的坑

- template<class T>T lookupType(const word&,bool recursive=false,bool patternMatch=true) const
- template<class T>T lookupOrDefault(const word&,const T&,bool recursive=false,bool patternMatch=true) const
- template<class T>T lookupOrAddDefault(const word&,const T&,bool recursive=false,bool patternMatch=true)
- template<class T>bool readIfPresent(const word&,T&,bool recursive=false,bool patternMatch=true) const
- const entry* lookupScopedEntryPtr(const word&,bool recursive,bool patternMatch) const
- bool isDict(const word&) const
- const dictionary* subDictPtr(const word&) const
- dictionary* subDictPtr(const word&)
- const dictionary& subDict(const word&) const
- dictionary& subDict(const word&)
- dictionary subOrEmptyDict(const word&,const bool mustRead = false) const
- const dictionary& optionalSubDict(const word&) const
- wordList toc() const
- wordList sortedToc() const
- List<keyType> keys(bool patterns=false) const

Editing

- bool substituteKeyword(const word& keyword)
- bool substituteScopedKeyword(const word& keyword)
- bool add(entry*, bool mergeEntry=false)
- void add(const entry&, bool mergeEntry=false)
- void add(const keyType&, const word&, bool overwrite=false)
- void add(const keyType&, const string&, bool overwrite=false)
- void add(const keyType&, const label, bool overwrite=false)
- void add(const keyType&, const scalar, bool overwrite=false)
- void add(const keyType&,const dictionary&,bool mergeEntry=false)
- template<class T>void add(const keyType&, const T&, bool overwrite=false)
- void set(entry*)
- void set(const entry&)
- void set(const keyType&, const dictionary&)
- template<class T>void set(const keyType&, const T&);
- bool remove(const word&)
- bool changeKeyword(const keyType& oldKeyword,const keyType& newKeyword,bool forceOverwrite=false)
- bool merge(const dictionary&)
- void clear()
- void transfer(dictionary&)
- Xfer<dictionary> xfer()

Read

- bool read(Istream&);
- bool read(Istream&, const bool keepHeader)

Write

- void write(Ostream&, const bool subDict=true) const

Member Operators

- ITstream& operator[] (const word&) const
- void operator=(const dictionary&)
- void operator+=(const dictionary&)
- void operator|=(const dictionary&)
- void operator<=<=(const dictionary&)

Iostream Operators

- friend Istream& operator>>(Istream&, dictionary&)
- friend Ostream& operator<<(Ostream&, const dictionary&)

Global Operators

- dictionary operator+(const dictionary& dict1, const dictionary& dict2)
- dictionary operator|(const dictionary& dict1, const dictionary& dict2)

C regIObject

C baseIOdictionary

Private data

- static bool writeDictionaries

Constructors

- baseIOdictionary(const IOobject&)
- baseIOdictionary(const IOobject&, const dictionary&)
- baseIOdictionary(const IOobject&, Istream&)

Member functions

- virtual fileName filePath() const = 0
- const word& name() const
- virtual bool readData(Istream&)
- virtual bool writeData(Ostream&) const
- virtual bool global() const = 0

Member operators

- void operator=(const baseIOdictionary&)

C IOdictionary

Constructors

- IOdictionary(const IOobject&);
- IOdictionary(const IOobject&, const dictionary&);
- IOdictionary(const IOobject&, Istream&);

Member functions

- virtual bool global() const{return true;}
- virtual fileName filePath() const{return globalFilePath(type());}

一些函数探秘

算例里的文件片段：

```
C++
1  RanzMarshallCoeffs
2  {
3      BirdCorrection  true;
4  }
```



目录

你已经读了 0 %

1. 常见用法
2. 代码解析
3. 踩过的坑

代码:

C++

```
1  const dictionary& subModelDict =  
    parcels.subModelProperty().subDict("RanzMarshallCoeffs");  
2  Info<<"RanzMarshallCoeffs.toc()=== "<<subModelDict.toc()<<endl;  
3  Info<<"RanzMarshallCoeffs.keys()=== "<<subModelDict.keys()<<endl;
```

输出:

C++

```
1  RanzMarshallCoeffs.toc()===1(BirdCorrection)  
2  RanzMarshallCoeffs.keys()===1(BirdCorrection)
```

算例里的文件片段:

C++

```
1  liquids  
2  {  
3      C7H16  
4      {  
5          defaultCoeffs    yes;  
6      }  
7      IC8H18  
8      {  
9          defaultCoeffs    yes;  
10     }  
11 }
```

代码:

C++

```
1  const wordList fuelList(thermo.subDict("liquids").keys());  
2  const wordList fuelCoeff1(thermo.subDict("liquids").subDict("C7H16").keys());  
3  Info<<"fuelList=== "<<fuelList<<endl;  
4  Info<<"fuelCoeff1=== "<<fuelCoeff1<<endl;
```

输出:

C++

```
1  fuelList===  
2  2  
3  (  
4  C7H16  
5  IC8H18  
6  )  
7  
8  fuelCoeff1===1(defaultCoeffs)
```

算例里的文件片段:

C++

```
1  Cd    constant 0.88;
```

目录

你已经读了 0 %

1. 常见用法
2. 代码解析
3. 踩过的坑

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++

```
1 scalar Cd_ = 0.;
2 token firstToken(modelDict.lookup("Cd")[1]);
3 //这里 lookup 得到一个list, 我们取[1], 即后边的 0.88
4 if (firstToken.isScalar())
5 {
6     Cd_ = firstToken.scalarToken();
7 }
```

其实“更”正确的用法是：

C++

```
1 TimeFunction1<scalar> Cd2_(mesh.time(),"Cd");
2 Cd2_.reset(modelDict);
3 Info<<"Cd2_=="<<Cd2_.value(0.)<<endl;
4 scalar Cd_ = Cd2_.value(0.);
```

算例里的文件片段：

C++

```
1 37 liquids
2 38 {
3 39     C7H16
4 40     {
5 41         defaultCoeffs yes;
6 42     }
7 43     IC8H18
8 44     {
9 45         defaultCoeffs yes;
10 46     }
11 47 }
```

代码：

C++

```
1 Info<<"thermo.subDict(liquids).subDict(C7H16).parent()=="
<<thermo.subDict("liquids").subDict("C7H16").parent()<<endl;
2 Info<<"thermo.subDict(liquids).topDict()=="<<thermo.subDict("liquids").topDict()<<endl;
3 Info<<"thermo.subDict(liquids).startLineNumber()=="<<thermo.subDict("liquids").startLineNumber()<<endl;
4 Info<<"thermo.subDict(liquids).endLineNumber()=="<<thermo.subDict("liquids").endLineNumber()<<endl;
5 Info<<"thermo.subDict(liquids).tokens()=="<<thermo.subDict("liquids").tokens()<<endl;
6 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++

```
1 thermo.subDict(liquids).startLineNumber()===41
2 thermo.subDict(liquids).endLineNumber()===45
3 thermo.subDict(liquids).tokens()===
4 12
```




目录

你已经读了 0 %

1. 常见用法
2. 代码解析
3. 踩过的坑

```

5  (
6  C7H16
7  {
8  defaultCoeffs
9  yes
10 ;
11 }
12 IC8H18
13 {
14 defaultCoeffs
15 yes
16 ;
17 }
18 )
19
20 thermo.subDict(liquids).toc()===
21 2
22 (
23 C7H16
24 IC8H18
25 )

```

`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++

```

1  Foam::wordList Foam::dictionary::toc() const
2  {
3      wordList keys(size());
4
5      label nKeys = 0;
6      forAllConstIter(IDLList<entry>, *this, iter)
7      {
8          keys[nKeys++] = iter().keyword();
9      }
10
11     return keys;
12 }
13
14 Foam::List<Foam::keyType> Foam::dictionary::keys(bool patterns=false) const
15 {
16     List<keyType> keys(size());
17
18     label nKeys = 0;
19     forAllConstIter(IDLList<entry>, *this, iter)
20     {
21         if (iter().keyword().isPattern() ? patterns : !patterns)
22         {
23             keys[nKeys++] = iter().keyword();
24         }
25     }
26     keys.setSize(nKeys);
27
28     return keys;
29 }

```

踩过的坑

我想读取一个文件，但是并行时只想让 `master` 读取。

初次尝试

首先构造一个 `IOdictionary` , `NO_READ`。

plain

```

1  IOdictionary Dict
2  (
3      IOobject
4      (
5          "dummyName", //tableName
6          path,
7          mesh,
8          IOobject::NO_READ,
9          IOobject::NO_WRITE
10     )
11 );

```

然后在 master 读取

plain

```

1  if (onlyMasterRead&&Pstream::parRun())
2  {
3      if (Pstream::master())
4      {
5          Dict.readOpt() = IOobject::MUST_READ;
6          Dict.readData(this->readStream(typeName));
7          Dict.close();
8      }
9  }
10 else
11 {
12     Dict.readOpt() = IOobject::MUST_READ;
13     Dict.readData(this->readStream(typeName));
14     Dict.close();
15 }

```

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  [1] --> FOAM FATAL IO ERROR:
2  [1] error in IOstream "IOstream" for operation operator>>(Istream&, List<T>&) :
   reading first token
3  [1]
4  [1] file: IOstream at line 0.
5  [1]
6  [1]     From function void Foam::IOstream::fatalCheck(const char*) const
7  [1]     in file db/IOstreams/IOstreams/IOstream.C at line 109.
8  [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`必须每个核保持一致，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

你已经读了 0 %

1. 常见用法
2. 代码解析
3. 踩过的坑



目录

你已经读了 0 %

1. 常见用法
2. 代码解析
3. 踩过的坑

```

1 plain IOdictionary Dict
2   (
3     IOobject
4     (
5       "dummyName", //tableName
6       path,
7       mesh,
8       IOobject::MUST_READ,
9       IOobject::NO_WRITE
10    )
11 );

```

然后在 master 读取真正的文件

plain

```

1   if (onlyMasterRead&&Pstream::parRun())
2   {
3       if (Pstream::master())
4       {
5           Dict.rename(tableName);
6           Dict.readData(this->readStream(typeName));
7           Dict.close();
8       }
9   }
10  else
11  {
12      Dict.rename(tableName);
13      Dict.readData(this->readStream(typeName));
14      Dict.close();
15  }

```

还是和上边一样的问题!

🔧 解决方案

不用 IOdictionary , 而是用 dictionary

plain

```

1   if (onlyMasterRead&&Pstream::parRun())
2   {
3       if (Pstream::master())
4       {
5           tableValues_ = dictionary(IFstream(tablePath+"/"+tableName)
6           ()).lookupOrDefault<List<scalar>>(tableName, defaultList);
7           Pstream::scatter(tableValues_);
8       }
9   }
10  else
11  {
12      tableValues_ = dictionary(IFstream(tablePath+"/"+tableName)
13      ()).lookupOrDefault<List<scalar>>(tableName, defaultList);
14  }

```

文章作者: Yan Zhang

文章链接: <https://openfoam.top/dictionary/>

版权声明: 本博客所有文章除特别声明外, 均采用 [CC BY-NC-SA 4.0](#) 许可协议。转载请注明来自 [OpenFOAM](#)

成长之路!



目录

你已经读了 0 %

- 1. 常见用法
- 2. 代码解析
- 3. 踩过的坑



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！