Fluent Mesh 网格文件说明文档

ł

作者：IforeverYH

目的：说明 Fluent Mesh 文件关于网格部分的数据存储格式；

版本：1.1；

更新说明：标准化文档格式；添加 Face Tree、Cell Tree 及 Hanging Nodes 的说明；添加二进制文件说明；

日期：2022 年 10 月 6 日

1 Fluent Mesh 网格文件简介

Fluent Mesh 文件是 Fluent 求解器用于计算的网格文件。其常用的文件后缀为 cas/msh。这两种文件后缀的关系是：msh 文件是 cas 文件的子集，仅包含 cas 文件中与网格相关的部分；cas 除网格内容外还涉及到求解器设置(超出本文的范围)。用户可根据需求选择文件后缀。

目前主流的 CFD 求解器 (STAR CCM+、OpenFOAM 等) 及网格生成器 (Fluent Meshing、Pointwise、ICEM 等) 都能读取或转化、生成 Fluent Mesh 网格文件。

表 1: Fluent Mesh 网格文件的基本特性

常用后缀	msh/cas
2D 网格	OK
3D 网格	OK
结构网格	OK
非结构网格	OK
多面体网格	OK
边界条件	OK
存储方式	ASCII/Binary

表 1介绍了 Fluent Mesh 网格文件的基本特性。网格文件支持 2D、3D 网格及多种单元类型；支持结构网格与非结构网格；支持 CutCell 切割体网格；可包含边界条件；可使用 ASCII 或 Binary 形式储存。

该文档参考 ANSYS Fluent User's Guide 19.2 版，未来版本更新可能与该版本内容不同。请注意自己使用的软件版本及需求。

下面将按 CFD 计算的一般需求对网格文件的内容进行详细说明。

2 ASCII 文件说明

ASCII 数据中有关控制及标号的数字默认为16 进制，有关数据值的浮点数或整型默认为10 进制，如有其他情况将在文中说明。

2.1 2D ASCII 文件示例

在开始格式说明前，先看一个简单的 2D 网格文本实例，以 2×2 网格为例，其形状如图 1 所示。图中的 n、f、c 分别代表 node、face、cell，其后的数字为相应元素的标号(用 16 进制表示)，该实例可对照第3章内容理解。

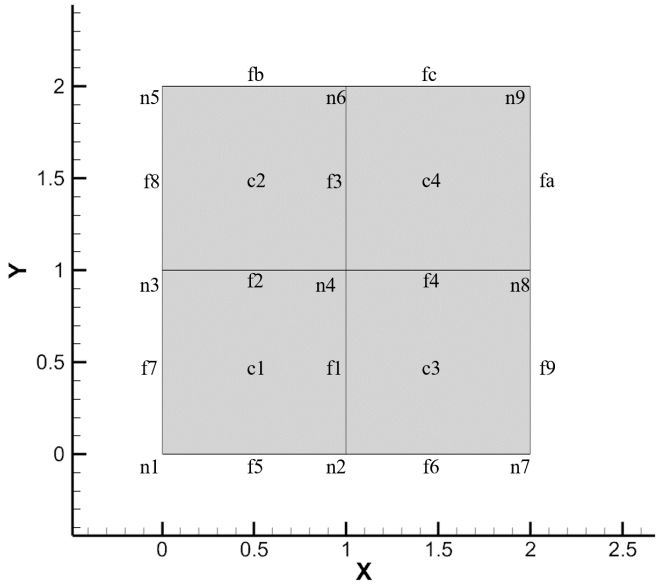

图 1: 2D 网格示例

```
(0 " Created by : Fluent_V6 Interface Vers. 18.2.0") % 版本信息
(2 2) ------------------------------------------- % 维度
(0 "Node Section") ---------------------------- % Nodes描述字段
(10 (0 1 9 0 2)) ------------------------------- % Nodes总述字段
(10 (5 1 9 1 2) ------------------------------- % Nodes头文字段
(0 0
1 0
0 1
1 1
0 2
1 2
2 0
2 1
2 2))
(12 (0 1 4 0 0)) ------------------------------- % Cells总述字段
(12 (6 1 4 1 3)) ------------------------------- % Cells头文字段
(13 (0 1 c 0 0)) ------------------------------- % Faces总述字段
(0 "Interior faces of zone FLUID") ------------- % 内部面描述字段
```

```
(13 (7 1 4 2 2)( ---------------------------------- % 内部Faces头文字段
2 4 1 3
4 3 1 2
4 6 2 4
8 4 3 4))
(0 "Faces of zone FAR") ---------------------------- % "far"面描述字段
(13 (8 5 c 9 2)( ---------------------------------- % "far"Faces头文字段
1 2 1 0
2 7 3 0
3 1 1 0
5 3 2 0
7 8 3 0
8 9 4 0
6 5 2 0
9 6 4 0))
(0 "Zone Sections") -------------------------------- % Zone描述字段
(39 (6 fluid FLUID)())
(39 (7 interior int_FLUID)())
(39 (8 pressure-far-field FAR)())
```

2.2 3D ASCII 文件示例

3D 文件的格式与 2D 相同，这里供保留部分内容做对比。网格以 $2 \times 2 \times 2$ 立方体为例，边长为 2，其形状如图 2 所示：

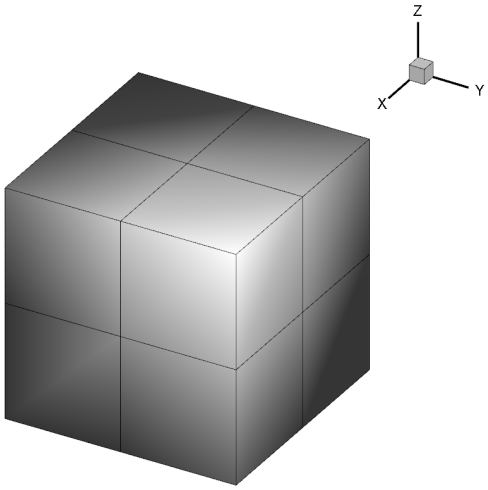

图 2: 3D 网格示例

```
(0 " Created by : Fluent_V6 Interface Vers. 18.2.0")
(2 3)
(0 "Node Section")
(10 (0 1 1b 0 3))
(10 (5 1 1b 1 3)(0 0 0 1 0 0...))
(12 (0 1 8 0 0))
```

```
(12 (6 1 8 1 4))
(13 (0 1 24 0 0))
(0 "Interior faces of zone FLUID")
(13 (7 1 c 2 4)(3 4 8 7 1 3 2 6 8 4 1 5...))
(0 "Faces of zone FAR")
(13 (1 d 24 9 4)(1 3 7 5 1 0 5 7 b 9 2 0...))
(0 "Zone Sections")
(39 (6 fluid FLUID)())
(39 (7 interior int_FLUID)())
(39 (1 pressure-far-field FAR)()) ------------ %自定义编号1
```

查看上方的例子后，可以总结的 ASCII 文件的特点，以及补充的特性说明：

1. Fluent Mesh 内容总是在成对"()" 内存放，形式是：(指令标号 + 内容)；
2. (指令标号 + 内容) 中的内容可以是"()" 的 list，list 内容的排布规则见下文；
3. 文档不要求特定位置的换行，即 (0 0 1 0 0 1...) 与 2D 例子 Nodes 的数据内容等价；也可参考 3D 例之中的写法理解；
4. 文档使用的分隔符为空格 (space)。

若按内容分类可分为：

1. **描述字段**：用来描述下方的内容，相当于注释(可省略)；
2. **总述字段**：用于对内容进行总述 (相当于头文字段的总结)，常见于节点，单元信息等模块(不可省略，且必须在对应的头文字段前)；
3. **头文字段**：用于描述各部分的内容的类型，成员起止编号，成员类型等信息(不可省略)；
4. **数据内容**：在相应的头文字段的"()" 内，为网格各类模块的具体信息，包括节点坐标，几何关系等内容(不可省略)。

3 Grid Sections

3.1 描述字段

```
{描述字段格式}: (0 "comment text")
{格式解释}: 0为指令标号;
            "comment text"为描述文本, 用""包括。
{实例}: (0 " Created by : Fluent_V6 Interface Vers. 18.2.0")
```

3.2 维度

```
{维度格式}: (2 n)
{格式解释}: 2为指令标号;
            第二个数字为维数, 2表示2维、3表示3维。
{实例}: (2 2); (2 3)
```

3.3 Nodes 节点

```
{节点格式}: (10 (zone-id first-index last-index type ND)(x1 y1 z1 x2 y2 z2... ))
{格式解释}: 10为指令标号;
            zone-id为区域编号;
            first-index为该区域第一个成员的编号;
            last-index为该区域最后一个成员的编号;
            type为节点类型;
            ND为维度;
            (x y z)为每个节点的坐标(10进制, Cartesian)。
{实例}: (10 (0 1 9 0 2)); (10 (5 1 9 1 2)(0 0 1 0 0 1....))
```

3.3.1 使用说明

1. **如果 zone-id = 0**：first-index = 1，last-index 为节点总数，type 置 0，ND 后面不跟坐标数据。此时相当于对 nodes 的整体说明，即总述字段；
2. **如果 zone-id > 0**：表示结构体中的 nodes 属于编号 zone-id 的 zone 区域。此时 first-index 和 last-index 为该 zone 区域的节点起止编号，节点的坐标信息在 ND 后的 () 内；
3. **如果 ND = 2**：表示 2 维，节点数据不包含 z 坐标。

3.3.2 type 设置规则

- 0: ”virtual” nodes – 虚拟节点；
- 1: no(any) type – 通用型；
- 2: boundary nodes – 边界节点。

3.4 Cells 单元

```
{单元格式}: (12 (zone-id first-index last-index type element-type)())
{格式解释}: 12为指令标号;
            zone-id为区域编号;
            first-index为该区域第一个成员的编号;
            last-index为该区域最后一个成员的编号;
            type为单元物理属性;
            element-type为单元类型;
            ()仅当element-type = 0时使用。
{实例}: (12 (0 1 4 0 0)); (12 (6 1 4 1 3))
```

3.4.1 使用说明

1. **如果 zone-id = 0**：first-index = 1，last-index 为单元总数 (若 last-index = 0 则表示文件中无 cell)，type 置 0，计算时 Fluent 会忽略 type = 0 的区域 (死区)，element-type 不显示 (或置 0)。此时相当于对 cells 的整体说明，即总述字段；

2. **如果 zone-id > 0**：表示结构体中的 cells 属于编号 zone-id 的 zone 区域。type = 1 为流体，type = 20 为 Cell Tree 的节点。

特别注意：如果单元类型为混合型 (element-type = 0)，则列出每个单元的类型。此时单元格式后接一对 ()。具体可见下面的例子，区域 9 有 61 个单元，其中前 3 个为三角单元，紧接两个六面体单元，等等。

```
(12 (9 1 3d 1 0)(
1 1 1 3 3 1 1 3 1 ...))
```

3.4.2 element-type 设置规则

- 0: mixed；
- 1: triangular – 三角形；
- 2: tetrahedral – 四面体；
- 3: quadrilateral – 四边形；
- 4: hexahedral – 六面体；
- 5: pyramid – 四棱锥（金字塔）；
- 6: wedge – 楔形；
- 7: polyhedral – 多面体。

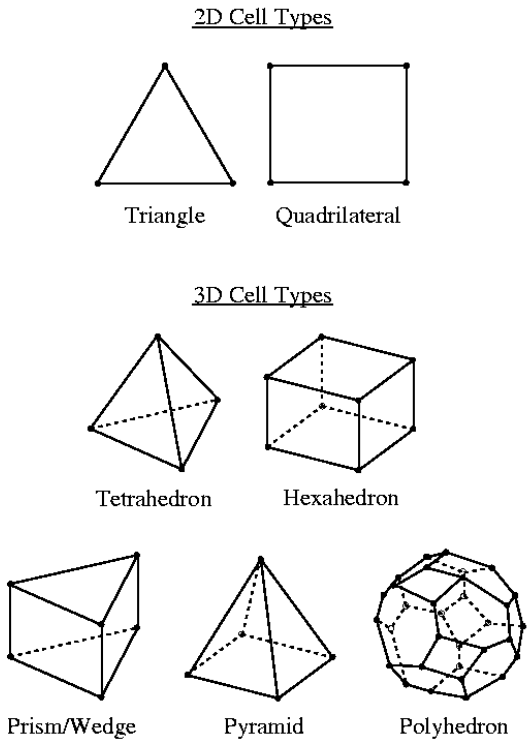

图 3: element-type 展示

3.5 Faces 面

```
{面格式}: (13 (zone-id first-index last-index bc-type face-type)())
{格式解释}: 13为指令标号;
            zone-id为区域编号;
            first-index为该区域第一个成员的编号;
            last-index为该区域最后一个成员的编号;
            bc-type为边界条件;
            face-type为面类型;
            ()内存放点、面、单元的关系。
{实例}: (13 (0 1 c 0 0)); (13 (7 1 4 2 2)(2 4 1 3 4 3 1 2...))
```

3.5.1 使用说明

1. **如果 zone-id = 0**：first-index = 1，last-index 为面总数，bc-type 不显示 (或置 0)。此时相当于对 faces 的整体说明，即总述字段；
2. **如果 zone-id > 0**：表示结构体中的 faces 属于编号 zone-id 的 zone 区域，面数据的信息在随后的 () 内。

3.5.2 bc-type 设置规则 (16 进制)

- `2(0x2)`: interior;
- `3(0x3)`: wall;
- `4(0x4)`: pressure-inlet, inlet-vent, intake-fan;
- `5(0x5)`: pressure-outlet, outlet-vent, exhaust-fan;
- `7(0x7)`: symmetry;
- `8(0x8)`: periodic-shadow;
- `9(0x9)`: pressure-far-field;
- `10(0xa)`: velocity-inlet;
- `12(0xc)`: periodic;
- `14(0xe)`: fan, porous-jump, radiztor;
- `20(0x14)`: mass-flow-inlet;
- `24(0x18)`: Interface;
- `31(0x1f)`: parent(hanging node);
- `36(0x24)`: outflow;
- `37(0x25)`: axis.

注意：非共形网格相交的交界面被存放在一个单独的面区域中。这些面区域的 bc-type 拓展 1000(例如 bc-type = 1003 是 wall)。

3.5.3 face-type 设置规则

- `0`: mixed;

- 2: linear;
- 3: triangular;
- 4: quadrilateral;
- 5: polygonal.

3.5.4 点、面、单元关系格式

```
{面关系格式}: (n0 n1 n2 c0 c1)
{格式解释}: n*表示节点编号，其最大标号与face-type相关;
            c*表示face的邻近cell编号，c0按右手法则确定，c1在face的另一边;
```

对于二维情况，假设存在一个指向平面外的 $\vec{k}$，通过 $\vec{k} \times \vec{r}$ 来确定 c0；在边界处 c0 或 c1 为 0。

当网格为混合类型 (face-type = 0) 或多边形类型 (face-type = 5)，每一行说明面的语句应以节点数目开头：

```
(x n0 n1 ... nf c0 c1)
{实例}: 五边形(... 5 89 8a 12 f 5 12 4 ...)
```

3.6 Face Tree

```
{区域格式}: (59 (face-id0 face-id1 parent-zone-id child-zone-id)
                (number-of-kids kid-id-0 kid-id-1 ... kid-id-n))
{格式解释}: 59为指令标号;
            face-id0为区域第一个父级面的编号;
            face-id1为区域最后一个父级面的编号;
            parent-zone-id为父级面所在区域id;
            child-zone-id为子级面所在区域id;
            number-of-kids为子级面的数量;
            kid-id-n为子级面的编号。
{实例}:  (59 (139 13a 5 9)(2 135 136 2 137 138))
```

该部分的详细网格实例可参考 5.1。

3.7 Cell Tree

```
{区域格式}: (58 (cell-id0 cell-id1 parent-zone-id child-zone-id)
                (number-of-kids kid-id-0 kid-id-1 ... kid-id-n))
{格式解释}: 58为指令标号;
            cell-id0为区域第一个父级单元的编号;
            cell-id1为区域最后一个父级单元的编号;
            parent-zone-id为父级单元所在区域id;
            child-zone-id为子级单元所在区域id;
            number-of-kids为子级单元的数量;
            kid-id-n为子级单元的编号。
```

```
{实例}: (58 (8d 92 2 6)(4 75 76 77 78 4 79 7a 7b 7c ...))
```

该部分的详细网格实例可参考 5.1。

4 Non-Grid Sections

4.1 Zone 区域

```
{区域格式}: (39/45 (zone-id zone-type zone-name domain-id)())
{格式解释}: 39/45为指令标号;
            zone-id为区域编号(10进制);
            zone-type为区域类型(一般是指边界条件);
            zone-name为区域名称;
            domain-id用于关联边界条件与相(整型);
            ()空。
{实例}: (39 (8 pressure-far-field FAR)())
```

需要注意的事项：

1. 头文字段最后的空”()” 必须存在，这部分内容由 Fluent 求解器填写;
2. 当 zone 内容只有 zone-id, zone-type, zone-name, 和 domain-id 时推荐指令标号 45;
3. 当有边界条件时指令标号必须用 39。

4.1.1 zone-type 设置规则

表 2: 可用的 zone-type 种类

degassing	exhaust-fan	fan
fluid	geometry	inlet-vent
intake-fan	interface	interior
internal	mass-flow-inlet	outflow
outlet-vent	parent-face	porous-jump
pressure-far-field	pressure-inlet	pressure-outlet
radiator	solid	symmetry
velocity-inlet	wall	wrapper

5 复杂情况的网格实例

5.1 Cartesian mesh

这里解释的 Cartesian mesh 是基于树形数据结构的笛卡尔网格，这种网格的特点之一是含有 Hanging Node。Hanging Node 指的是拓扑结构不完备的 Node，如图4左中圈出的 Node 不属于其下方的三角形；与之相对的是右中展示的网格，该网格中不含有 Hanging Node。

目前，Cartesian mesh 常用于 AMR(Adaptive Mesh Refinement) 自适应网格方法中的网格重构，以及对复杂曲面的边界加密或跟踪。主流的 CFD 求解器都有相应 Cartesian mesh 生成方法 (如 Fluent Meshing 的 CutCell 网格生成器、STAR CCM+ 的 Trimmed Mesher 网格生成器、以及 OpenFOAM 的 SnppyHexMesh 网格生成器)。

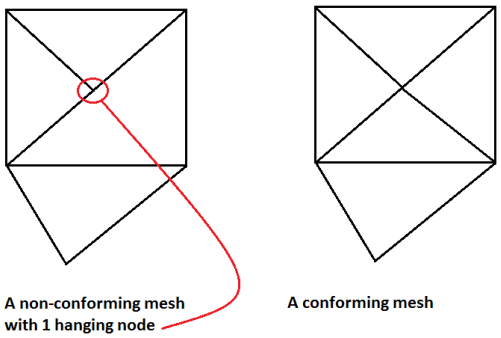

图 4: Hanging Node 示例

在 Fluent 求解器中 Haning Nodes 由 Face Tree、Cell Tree 控制，这两个区域描述的内容类似树形结构 (Tree) 的节点，一层层嵌入网格信息。这里采用 Fluent Mesh 中树形 Cartesian mesh 结构，给出一层 Tree 结构的网格实例 (多层结构类似)。可参考 Face Tree 3.6，及 Cell Tree 3.7理解相应内容，由于 Nodes、Faces、cells 都在图中一一标明，相应内容不再赘述，这个实例的关注点在 Face Tree 和 Cell Tree 的结构上。具体可参考图 6。

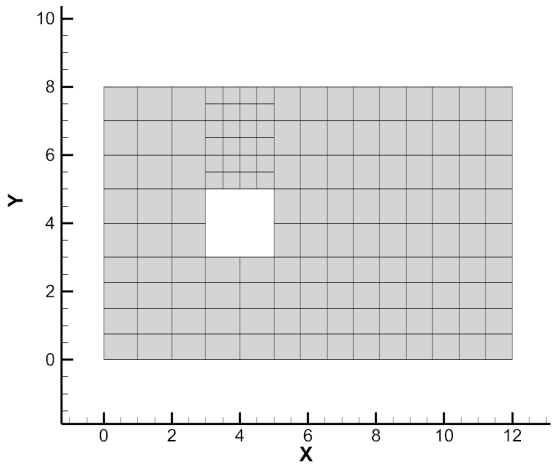

图 5: 树形数据结构 Cartesian mesh 示例

```
(0 "Grid:")
(12 (0 1 92 0))
(13 (0 1 149 0))
(10 (0 1 ac 0 2))
(12 (6 1 8c 1 3))
(12 (2 8d 92 20 3)) -------------- % Cell的type为20，表示parent-cell
(58 (8d 92 2 6) ----------------- % Cell Tree 其内容可理解为树形结构中节点的概念
(4 75 76 77 78
```

```
 4 79 7a 7b 7c
 4 7d 7e 7f 80
 4 81 82 83 84
 4 85 86 87 88
 4 89 8a 8b 8c))
(13 (7 1 fe 2 2)(1 2 1 5 2 3 1 2 2 4 2 6 4 5 2 3 ...))
(13 (8 ff 12e 9 2)(a9 1 1 0 3 a9 1 0 5 3 2 0 7 5 3 ...))
(13 (9 12f 138 3 2)(13 1a 11 0 1a 1b 12 0 2a 13 1f 0 ...))
(13 (5 139 13a 1f 2) ------------- % Face的bc-type为1f, 表示parent-face, 下同
(1b 8c 8d 0
8c 5f 90 0))
(13 (4 13b 13c 1f 2)
(99 9b 8f 0
a6 99 92 0))
(13 (3 13d 149 1f 2)
(8c 8e 8d 90
8e 24 8d 8e
1b 24 19 8d
8e 94 8e 91
94 25 8e 8f
24 25 1a 8e
94 99 8f 92
25 9b 1b 8f
5f 71 90 5a
71 8e 90 91
71 73 91 5b
73 94 91 92
73 a6 92 5c))
(59 (139 13a 5 9)
(2 135 136
2 137 138))
(59 (13b 13c 4 8)
(2 12b 12c
2 12d 12e))
(59 (13d 149 3 7)
(2 cd ce
2 cf d0
2 d1 d2
2 d7 d8
2 d9 da
2 db dc
2 e1 e2
2 e3 e4
2 e9 ea
2 eb ec
2 f1 f2
2 f3 f4
```

```
2 f9 fa))
(10 (1 1 ac 1 2)(1.0000000000000000e+00  0.0000000000000000e+00 ...))
(39 (6 fluid fluid 1)())
(39 (8 pressure-far-field far 1)())
(39 (7 interior int_fluid 1)())
(39 (9 wall wall 1)())
```

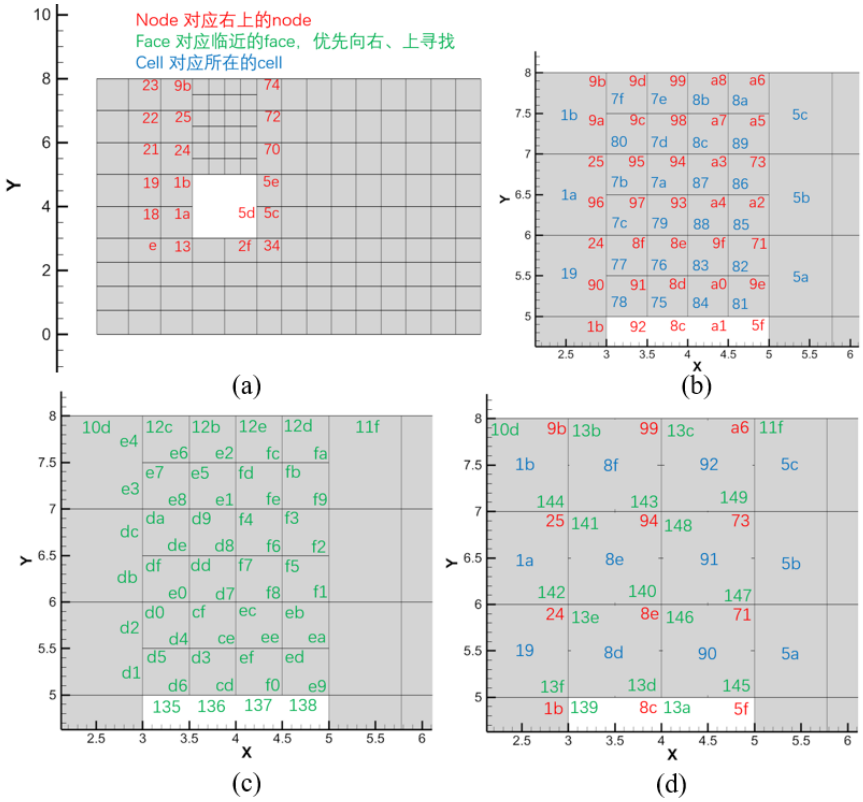

图 6: 网格元素标号图；(a) 网格总体视图；(b)Tree 区域的 child-cells、nodes 标号；(c)Tree 区域的 child-faces 标号；(d)Tree 区域的 parent-cells、faces、nodes 标号

需要注意的事项：

1. 可参考图7的内容理解何为网格 tree 结构；
2. Fluent 中的 Face、Cell Tree 分别用来确定存在 child-face、cell 的 Face 和 Cell，各层级间不允许跨越，各层的网格必须一层层排列。安全的做法就是一层一层写，不要考虑的很复杂；
3. 通过静态网格生成器生成的网格可能缺失 tree 结构的信息。

6 二进制文件说明

Fluent Mesh 的二进制文件与 ASCII 文件的区别非常小，要说明的是二进制文件除了数据内容外的内容都按 ASCII 解码。下面是一段二进制文件的实例。

```
{实例}: (3013 (7 d 10 5 4)(二进制数据内容)End of Binary Section 3013)
```

需要说明的事项：

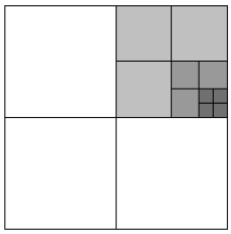

Figure 1: a recursively divided region, colored according to depth.

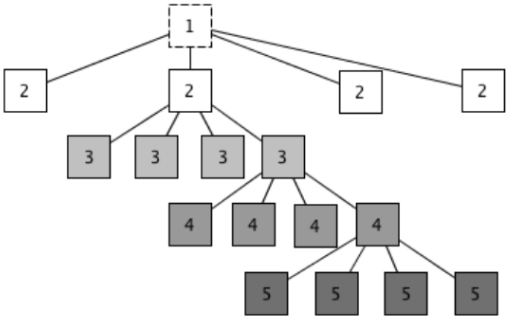

Figure 2: a quadtree for Figure 1.

图 7: 网格 Tree 结构说明图

1. 二进制文件中除数据内容外均按 ASCII 存储；
2. 对于有数据内容的区域，二进制的头文字段与 ASCII 文件相比，指令标号增加两个拓展位 (20 为单精度；30 为双精度)
3. 二进制文件的数据以二进制存储，每个数据占用的字节数由指令标号添加的拓展位确定，数据间不需要分隔；
4. 二进制数据结尾以指定的字符串标识”End of Binary Section xxxx”。

注：Cell 区域有时数据为空 (不等于无数据，空就是数据)，也要按二进制文件格式在结尾添加指定的字符串标识。

7 附录

7.1 Index 检索

检索列表列出 Fluent Mesh 文件网格内容的所有可用的指令标号，后接-optional 的代表可选项目；后接-required 的代表必需项目或依照相应条件的必需项目。

- `0`: Comment[3.1], -optional;
- `1`: Header, -optional;
- `2`: Dimensions[3.2], -optional;
- `10`: Nodes[3.3], -required;
- `11`: Edges, -optional;
- `12`: Cells[3.4], -required;
- `13`: Faces[3.5], -required;
- `18`: Periodic Shadow Faces, -required with periodic boundaries;
- `39 or 45`: Zone[4.1], -required;(当有边界条件时必须用 39)
- `58`: Cell Tree[3.7], -required only for grids with hanging-node adaption;
- `59`: Face Tree[3.6], -required only for grids with hanging-node adaption;
- `61`: Interface Face Parents, -required only for grids with non-conformal Interfaces.

7.2 X-type 设置检索

1. node-type[3.3.2];
2. element-type[3.4.2];
3. bc-type[3.5.2];
4. face-type[3.5.3];
5. zone-type[4.1.1].

7.3 文档资源

如果需要文档的源码、图片资源请到 GitHub 项目页面 下载。