Hierarchical Data Management library

0.3.0

構築: Doxygen 1.8.6

2014 年 03 月 29 日 (土) 18 時 13 分 04 秒

Contents

Chapter 1

名前空間索引

1.1 名前空間一覧

全名前空間の一覧です。

Chapter 2

クラス索引

2.1 クラス一覧

クラス・構造体・共用体・インターフェースの一覧です。

Chapter 3

ファイル索引

3.1 ファイル一覧

ファイル一覧です。

Chapter 4

名前空間詳解

4.1 BCMFileIO 名前空間

クラス

- struct OctHeader
  Octree **ファイルヘッダ構造体**
- struct LBHeader
  LeafBlock **ファイルヘッダ構造体**
- struct LBCellIDHeader
  LeafBlock **の***CellID* **ヘッダ構造体**
- struct GridRleCode
  RLE **圧縮符号の走査用構造体**
- struct IdxUnit
  インデックスファイル用単位系情報
- struct IdxProc
  インデックスファイル用プロセス情報
- class BCMFileLoader
  BCM **ファイルを読み込むクラス**
- class BCMFileSaver
  BCM **ファイルを出力するクラス**
- class BCMRLE
  ランレングスによる圧縮/展開ライブラリ
- class BitVoxel
  ビットボクセル圧縮/展開ライブラリ
- class DirUtil
  ディレクトリ操作ユーティリティ
- class ErrorUtil
  エラー処理関連のユーティリティ
- class FileSystemUtil
  ファイル操作関連ユーティリティ
- class IdxBlock
  インデックスファイル用ブロック情報クラス
- class IdxStep
  インデックスファイル用タイムステップ情報

- class LeafBlockLoader

  LeafBlock **ファイルを読み込むクラス**
- class LeafBlockSaver

  LeafBlock **ファイルを出力する関数群**
- class Logger

  ログ出力ユーティリティ
- class PartitionMapper

  MxN **データロードのためのマッピングクラス**

型定義

- typedef
  LeafBlockLoader::CellIDCapsule CellIDCapsule
- typedef unsigned int bitVoxelCell

列挙型

- enum LB_KIND {
  LB_CELLID = 0, LB_SCALAR = 1, LB_VECTOR3 = 3, LB_VECTOR4 = 4,
  LB_VECTOR6 = 6, LB_TENSOR = 9 }

  リーフブロックデータタイプ
- enum LB_DATA_TYPE {
  LB_INT8 = 0, LB_UINT8 = 1, LB_INT16 = 2, LB_UINT16 = 3,
  LB_INT32 = 4, LB_UINT32 = 5, LB_INT64 = 6, LB_UINT64 = 7,
  LB_FLOAT32 = 8, LB_FLOAT64 = 9 }

  リーフセルのデータ識別子

関数

- void DUMMY (void ∗)
- static void BSwap16 (void ∗a)

  2byte **用エンディアンスワップ**
- static void BSwap32 (void ∗a)

  4byte **用エンディアンスワップ**
- static void BSwap64 (void ∗a)

  8byte **用エンディアンスワップ**

変数

- struct BCMFileIO::OctHeader ALIGNMENT

4.1.1 型定義詳解

4.1.1.1 typedef BitVoxel::bitVoxelCell BCMFileIO::bitVoxelCell

BitVoxel.cpp の 20 行目に定義があります。

4.1.1.2 typedef LeafBlockLoader::CellIDCapsule BCMFileIO::CellIDCapsule

BCMFileLoader.cpp の 49 行目に定義があります。

4.1.2 列挙型詳解

4.1.2.1 enum BCMFileIO::LB_DATA_TYPE

リーフセルのデータ識別子

列挙値

LB_INT8 符号付き 8bit 整数型
LB_UINT8 符号なし 8bit 整数型
LB_INT16 符号付き 16bit 整数型
LB_UINT16 符号なし 16bit 整数型
LB_INT32 符号付き 32bit 整数型
LB_UINT32 符号なし 32bit 整数型
LB_INT64 符号付き 64bit 整数型
LB_UINT64 符号なし 64bit 整数型
LB_FLOAT32 32bit 浮動小数点 (単精度浮動小数点)
LB_FLOAT64 64bit 浮動小数点 (倍精度浮動小数点)

BCMFileCommon.h の 113 行目に定義があります。

```
114         {
115                 LB_INT8    =  0,
116                 LB_UINT8   =  1,
117                 LB_INT16   =  2,
118                 LB_UINT16  =  3,
119                 LB_INT32   =  4,
120                 LB_UINT32  =  5,
121                 LB_INT64   =  6,
122                 LB_UINT64  =  7,
123                 LB_FLOAT32 =  8,
124                 LB_FLOAT64 =  9
125         };
```

4.1.2.2 enum BCMFileIO::LB_KIND

リーフブロックデータタイプ

列挙値

LB_CELLID グリッド
LB_SCALAR スカラ
LB_VECTOR3 ベクトル (3 要素)
LB_VECTOR4 ベクトル (4 要素)
LB_VECTOR6 ベクトル (6 要素)
LB_TENSOR テンソル (9 要素)

BCMFileCommon.h の 102 行目に定義があります。

```
103         {
104                 LB_CELLID  = 0,
105                 LB_SCALAR  = 1,
106                 LB_VECTOR3 = 3,
107                 LB_VECTOR4 = 4,
108                 LB_VECTOR6 = 6,
109                 LB_TENSOR  = 9,
110         };
```

4.1.3 関数詳解

4.1.3.1 static void BCMFileIO::BSwap16 (void * *a*) [inline], [static]

2byte 用エンディアンスワップ

BCMFileCommon.h の 146 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::Load_BlockContents(), BCMFileIO::LeafBlockLoader::LoadHeader().

```
146                                         {
147                 unsigned short* x = (unsigned short*)a;
148                 *x = (unsigned short)( ((((*x) & 0xff00) >> 8 ) | (((*x) & 0x00ff) << 8)) );
149         }
```

4.1.3.2 static void BCMFileIO::BSwap32 (void * *a*) [inline], [static]

4byte 用エンディアンスワップ

BCMFileCommon.h の 152 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::Load_BlockContents(), BCMFileIO::LeafBlockLoader::LoadCellIDData(), BCMFileIO::LeafBlockLoader::LoadHeader(), BCMFileIO::BCMFileLoader::LoadOctreeHeader().

```
152                                         {
153                 unsigned int* x = (unsigned int*)a;
154                 *x = ( (((*x) & 0xff000000) >> 24) | (((*x) & 0x00ff0000) >> 8 ) |
155                        (((*x) & 0x0000ff00) <<  8) | (((*x) & 0x000000ff) << 24) );
156         }
```

4.1.3.3 static void BCMFileIO::BSwap64 (void * *a*) [inline], [static]

8byte 用エンディアンスワップ

BCMFileCommon.h の 159 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::Load_BlockContents(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadCellIDHeader(), BCMFileIO::LeafBlockLoader::LoadHeader(), BCMFileIO::BCMFileLoader::LoadOctreeFile(), BCMFileIO::BCMFileLoader::LoadOctreeHeader().

```
159                                         {
160                 uint64_t* x = (uint64_t*)a;
161                 *x=( (((*x) & 0xff00000000000000ull) >> 56) | (((*x) & 0x00ff000000000000ull) >> 40) |
162                      (((*x) & 0x0000ff0000000000ull) >> 24) | (((*x) & 0x000000ff00000000ull) >>  8) |
163                      (((*x) & 0x00000000ff000000ull) <<  8) | (((*x) & 0x0000000000ff0000ull) << 24) |
164                      (((*x) & 0x000000000000ff00ull) << 40) | (((*x) & 0x00000000000000ffull) << 56) );
165         }
```

4.1.3.4 void BCMFileIO::DUMMY (void *) [inline]

LeafBlockLoader.cpp の 32 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::Load_BlockContents().

```
32 {}
```

4.1.4 変数詳解

4.1.4.1 struct BCMFileIO::GridRleCode BCMFileIO::ALIGNMENT

参照元 BCMFileIO::BCMRLE::Decode(), BCMFileIO::BCMRLE::Encode().

4.2 Vec3class 名前空間

クラス

- class Vec3

型定義

- typedef Vec3< unsigned char > Vec3uc
- typedef Vec3< int > Vec3i
- typedef Vec3< float > Vec3f
- typedef Vec3< double > Vec3d

列挙型

- enum AxisEnum { AXIS_X = 0, AXIS_Y, AXIS_Z, AXIS_ERROR }

関数

- template<typename T >
  Vec3< T > operator* (T s, const Vec3< T > &v)
- template<typename T >
  Vec3< T > multi (const Vec3< T > &a, const Vec3< T > &b)
- template<typename T >
  T dot (const Vec3< T > &a, const Vec3< T > &b)
- template<typename T >
  Vec3< T > cross (const Vec3< T > &a, const Vec3< T > &b)
- template<typename T >
  T distanceSquared (const Vec3< T > &a, const Vec3< T > &b)
- template<typename T >
  T distance (const Vec3< T > &a, const Vec3< T > &b)
- bool lessVec3f (const Vec3f &a, const Vec3f &b)
- template<typename T >
  std::istream & operator>> (std::istream &is, Vec3< T > &v)

- template<typename T >
  std::ostream & operator<< (std::ostream &os, const Vec3< T > &v)
- std::istream & operator>> (std::istream &is, Vec3uc &v)
- std::ostream & operator<< (std::ostream &os, const Vec3uc &v)

4.2.1 型定義詳解

4.2.1.1 typedef Vec3<double> Vec3class::Vec3d

Vec3.h の 186 行目に定義があります。

4.2.1.2 typedef Vec3<float> Vec3class::Vec3f

Vec3.h の 185 行目に定義があります。

4.2.1.3 typedef Vec3<int> Vec3class::Vec3i

Vec3.h の 184 行目に定義があります。

4.2.1.4 typedef Vec3<unsigned char> Vec3class::Vec3uc

Vec3.h の 183 行目に定義があります。

4.2.2 列挙型詳解

4.2.2.1 enum Vec3class::AxisEnum

列挙値

AXIS_X

AXIS_Y

AXIS_Z

AXIS_ERROR

Vec3.h の 39 行目に定義があります。

```
39             {
40         AXIS_X = 0,
41         AXIS_Y,
42         AXIS_Z,
43         AXIS_ERROR
44 } AxisEnum;
```

4.2.3 関数詳解

4.2.3.1 template<typename T > Vec3<T> Vec3class::cross (const Vec3< T > & *a*, const Vec3< T > & *b*) [inline]

Vec3.h の 208 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
208                                                     {
209         return Vec3<T>(a.t[1] * b.t[2] - a.t[2] * b.t[1],
210                 a.t[2] * b.t[0] - a.t[0] * b.t[2],
211                 a.t[0] * b.t[1] - a.t[1] * b.t[0]);
212 }
```

4.2.3.2 template<typename T > T Vec3class::distance (const Vec3< T > & *a*, const Vec3< T > & *b*) [inline]

Vec3.h の 220 行目に定義があります。

```
220                                                     {
221         return (a - b).length();
222 }
```

4.2.3.3 template<typename T > T Vec3class::distanceSquared (const Vec3< T > & *a*, const Vec3< T > & *b*) [inline]

Vec3.h の 215 行目に定義があります。

```
215                                                         {
216         return (a - b).lengthSquared();
217 }
```

4.2.3.4 template<typename T > T Vec3class::dot (const Vec3< T > & *a*, const Vec3< T > & *b*) [inline]

Vec3.h の 203 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
203                                                 {
204         return a.t[0] * b.t[0] + a.t[1] * b.t[1] + a.t[2] * b.t[2];
205 }
```

4.2.3.5 bool Vec3class::lessVec3f (const Vec3f & *a*, const Vec3f & *b*) [inline]

Vec3.h の 225 行目に定義があります。

参照先 Vec3class::Vec3< T >::lengthSquared().

```
226 {
227         return (a.lengthSquared() < b.lengthSquared()) ? true : false;
228 }
```

4.2.3.6 template<typename T > Vec3<T> Vec3class::multi (const Vec3< T > & *a*, const Vec3< T > & *b*) [inline]

Vec3.h の 198 行目に定義があります。

```
198                                                     {
199         return a * b;
200 }
```

4.2.3.7 template<typename T > Vec3<T> Vec3class::operator∗ (T *s*, const Vec3< T > & *v*) [inline]

Vec3.h の 193 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
193                                             {
194         return Vec3<T>(s*v.t[0], s*v.t[1], s*v.t[2]);
195 }
```

4.2.3.8 template<typename T > std::ostream& Vec3class::operator<< (std::ostream & *os*, const Vec3< T > & *v*) [inline]

Vec3.h の 240 行目に定義があります。

```
241 {
242         return os << v[0] << " " << v[1] << " " << v[2];
243 }
```

4.2.3.9 std::ostream& Vec3class::operator<< (std::ostream & *os*, const Vec3uc & *v*) [inline]

Vec3.h の 255 行目に定義があります。

```
256 {
257         int x[3];
258         x[0]=v[0]; x[1]=v[1]; x[2]=v[2];
259         os << x[0] << " " << x[1] << " " << x[2];
260         return os;
261 }
```

4.2.3.10 template<typename T > std::istream& Vec3class::operator>> (std::istream & *is*, Vec3< T > & *v*) [inline]

Vec3.h の 234 行目に定義があります。

```
235 {
236         return is >> v[0] >> v[1] >> v[2];
237 }
```

4.2.3.11 std::istream& Vec3class::operator>> (std::istream & *is*, Vec3uc & *v*) [inline]

Vec3.h の 247 行目に定義があります。

```
248 {
249         int x[3];
250         is >> x[0] >> x[1] >> x[2];
251         v[0]=x[0]; v[1]=x[1]; v[2]=x[2];
252         return is;
253 }
```

Chapter 5

クラス詳解

5.1 BCMFileIO::BCMFileLoader クラス

BCM ファイルを読み込むクラス

`#include <BCMFileLoader.h>`

BCMFileIO::BCMFileLoader 連携図

公開メンバ関数

- BCMFileLoader (const std::string &idxFilename, BoundaryConditionSetterBase ∗bcsetter)
- ∼BCMFileLoader ()
- bool LoadAdditionalIndex (const std::string &filepath)
- bool CreateLeafBlock (int ∗dataClassID, const std::string &name, const unsigned int vc, const bool separateVCUpdate=false)
- bool LoadLeafBlock (int ∗dataClassID, const std::string &name, const unsigned int vc, const unsigned int step=0, const bool separateVCUpdate=false)
- BCMOctree ∗ GetOctree ()
- Vec3d GetGlobalOrigin () const
- Vec3d GetGlobalRegion () const
- const IdxStep ∗ GetStep (const std::string &name) const
- const IdxUnit & GetUnit () const

非公開メンバ関数

- bool LoadOctree (const std::string &filename, BoundaryConditionSetterBase ∗bcsetter)
- bool LoadIndex (const std::string &filename, Vec3d &globalOrigin, Vec3d &globalRegion, std::string &octreeFilename, Vec3i &blockSize, std::vector< IdxProc > &idxProcList, std::vector< IdxBlock > &idxBlockList)
- bool LoadIndexStep (TextParser ∗tp, IdxStep ∗step)
- bool LoadIndexData (TextParser ∗tp, IdxBlock ∗ib)
- bool LoadIndexCellID (TextParser ∗tp, IdxBlock ∗ib)
- bool LoadIndexProc (const std::string &filename, std::vector< IdxProc > &procList)
- void PrintIdxInformation ()

  読み込んだインデックスファイルの内容を *stdout* に出力
- void PrintOctreeInformation ()

  読み込んだ*Octree* ファイルの内容を *stdout* に出力

- int GetUniqueTag ()
  ユニークなタグを取得
- int CompStr (const std::string &str1, const std::string &str2, bool ignorecase=true)
- int ReadVec3 (TextParser *tp, const std::string &label, Vec3d &v)
- int ReadVec3 (TextParser *tp, const std::string &label, Vec3i &v)
- bool GetType (const std::string &typeStr, LB_DATA_TYPE &retType)
- bool LoadOctreeHeader (FILE *fp, OctHeader &header, bool &isNeedSwap)
- bool LoadOctreeFile (const std::string &filename, OctHeader &header, std::vector< Pedigree > &pedigrees)

非公開変数類

- BlockManager & m_blockManager
  ブロックマネージャ
- const MPI::Intracomm & m_comm
  MPI **コミュニケータ**
- Vec3i m_leafBlockSize
  リーフブロックサイズ
- std::vector< IdxProc > m_idxProcList
  プロセス情報リスト
- std::vector< IdxBlock > m_idxBlockList
  ブロック情報リスト
- IdxUnit m_unit
  単位系
- Vec3d m_globalOrigin
  領域全体の原点座標
- Vec3d m_globalRegion
  領域全体の長さ
- BCMOctree * m_octree
  Octree.
- PartitionMapper * m_pmapper
  MxN **データマッパ**

5.1.1 詳解

BCM ファイルを読み込むクラス

BCMFileLoader.h の 42 行目に定義があります。

5.1.2 構築子と解体子

5.1.2.1 BCMFileIO::BCMFileLoader::BCMFileLoader (const std::string & *idxFilename*, BoundaryConditionSetterBase * *bcsetter*)

コンストラクタ

引数

in	*idxFilepath*	入力インデックスファイル名 (実行ディレクトリからの相対パス)
in	*bcsetter*	境界条件設定クラス

BCMFileLoader.cpp の 51 行目に定義があります。

参照先 BCMFileIO::FileSystemUtil::ConvertPath(), BCMFileIO::Logger::Error(), BCMFileIO::FileSystemUtil::GetDirectory(), LoadIndex(), LoadOctree(), m_globalOrigin, m_globalRegion, m_idxBlockList, m_idxProcList, m_leafBlockSize, BCMFileIO::ErrorUtil::reduceError().

```
52          : m_blockManager(BlockManager::getInstance()),
53            m_comm(m_blockManager.getCommunicator()),
54            m_octree(NULL),
55            m_pmapper(NULL)
56        {
57
58                std::string dir = FileSystemUtil:: GetDirectory(
    FileSystemUtil::ConvertPath(idxFilename));
59
60                std::string octreeFilename;
61                if( ErrorUtil::reduceError( !LoadIndex(idxFilename,
    m_globalOrigin, m_globalRegion, octreeFilename,
62                                                    m_leafBlockSize,
    m_idxProcList, m_idxBlockList ) ) ){
63                        Logger::Error("load index file error (%s) [%s:%d].\n", idxFilename.
    c_str(), __FILE__, __LINE__);
64                        return;
65                }
66
67                if( ErrorUtil::reduceError( !LoadOctree(std::string(dir +
    octreeFilename), bcsetter) ) ){
68                        Logger::Error("load octree file error (%s) [%s:%d].\n", std::string(
    dir + octreeFilename).c_str(), __FILE__, __LINE__);
69                        return;
70                }
71        }
```

5.1.2.2 BCMFileIO::BCMFileLoader::~BCMFileLoader ( )

デストラクタ

覚え書き

ファイルから読み込んだOctree はデストラクタで解放される.

BCMFileLoader.cpp の 74 行目に定義があります。

参照先 m_octree, m_pmapper.

```
75        {
76                if(m_pmapper != NULL) delete m_pmapper;
77                if(m_octree  != NULL) delete m_octree;
78        }
```

5.1.3 関数詳解

5.1.3.1 int BCMFileIO::BCMFileLoader::CompStr (const std::string & *str1*, const std::string & *str2*, bool *ignorecase* = `true`) [inline], [private]

2 つの文字列を比較

引数

in	*str1*	文字列 1
in	*str2*	文字列 2
in	*ignorecase*	true: 大文字、小文字を区別せず比較、false: 大文字、小文字を区別して比較

戻り値

負: str1 < str2、ゼロ: str1 == str2、正: str1 > str2

BCMFileLoader.cpp の 834 行目に定義があります。

参照元 GetType(), LoadIndex(), LoadIndexCellID(), LoadIndexData(), LoadIndexProc(), LoadIndexStep().

```
835         {
836                 std::string lstr1 = str1;
837                 std::string lstr2 = str2;
838                 if( ignorecase ){
839                         std::transform(lstr1.begin(), lstr1.end(), lstr1.begin(), ::tolower);
840                         std::transform(lstr2.begin(), lstr2.end(), lstr2.begin(), ::tolower);
841                 }
842
843                 return lstr1.compare(lstr2);
844         }
```

5.1.3.2 bool BCMFileIO::BCMFileLoader::CreateLeafBlock (int * *dataClassID*, const std::string & *name*, const unsigned int *vc*, const bool *separateVCUpdate* = false)

ブロックを生成

引数

out	*dataClassID*	生成したブロックのデータクラスID (配列の先頭アドレス)
in	*name*	系の名称
in	*vc*	仮想セルサイズ
in	*separateVC-Update*	仮想セルの同期方法フラグ．true の場合、3 軸方向別々に同期を行う

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

name に対応するブロックがすでに生成されている場合、新しいブロックは生成せず、対応するデータクラスID を返す name に対応するデータが複数コンポーネント (Vertor/Tensor) の場合、dataClassID の

separateVCUpdate は，読み込むデータクラスに対する初回呼び出しのみ有効

BCMFileLoader.cpp の 617 行目に定義があります。

参照先 BCMFileIO::Logger::Error(), BCMFileIO::IdxBlock::find(), GetUniqueTag(), BCMFileIO::LB_CELLID, BCMFileIO::LB_FLOAT32, BCMFileIO::LB_FLOAT64, BCMFileIO::LB_INT16, BCMFileIO::LB_INT32, BCMFileIO::LB_INT64, BCMFileIO::LB_INT8, BCMFileIO::LB_UINT16, BCMFileIO::LB_UINT32, BCMFileIO::LB_UINT64, BCMFileIO::LB_UINT8, m_blockManager, m_idxBlockList, BCMFileIO::ErrorUtil::reduceError() (計 17 項目).

参照元 LoadLeafBlock().

```
618         {
619                 using namespace std;
```

```
620                 bool err = false;
621
622                 IdxBlock* ib = IdxBlock::find(m_idxBlockList, name);
623
624                 if( ib == NULL ){
625                         Logger::Error("No such name as \"%s\" in loaded index.[%s:%d]\n", name
    .c_str(), __FILE__, __LINE__);
626                         err = true;
627                 }
628                 if( ErrorUtil::reduceError(err) ) { return false; }
629
630                 if(ib->kind == LB_CELLID)
631                 {
632                         if( ib->dataClassID.size() == 0 ){
633                                 ib->dataClassID.resize(1);
634                                 ib->dataClassID[0] = m_blockManager.setDataClass<
    Scalar3D<unsigned char> >(vc); // TODO Cell Updater
635                         }
636                         dataClassID[0] = ib->dataClassID[0];
637                 }
638                 else
639                 {
640                         if( ib->dataClassID.size() == 0 ){
641                                 ib->dataClassID.resize(static_cast<size_t>(ib->kind));
642                                 for(int i = 0; i < static_cast<int>(ib->kind); i++){
643                                         if     (ib->dataType == LB_FLOAT32){ ib->dataClassID[i] =
     m_blockManager.setDataClass< Scalar3D<f32>, Scalar3DUpdater<f32> >(vc); }
644                                         else if(ib->dataType == LB_FLOAT64){ ib->dataClassID[i] =
     m_blockManager.setDataClass< Scalar3D<f64>, Scalar3DUpdater<f64> >(vc); }
645                                         #if 0 // TODO
646                                         else if(ib->dataType == LB_INT8   ){ ib->dataClassID[i] =
    m_blockManager.setDataClass< Scalar3D< s8>, Scalar3DUpdater< s8> >(vc); }
647                                         else if(ib->dataType == LB_UINT8  ){ ib->dataClassID[i] =
    m_blockManager.setDataClass< Scalar3D< u8>, Scalar3DUpdater< u8> >(vc); }
648                                         else if(ib->dataType == LB_INT16  ){ ib->dataClassID[i] =
    m_blockManager.setDataClass< Scalar3D<s16>, Scalar3DUpdater<s16> >(vc); }
649                                         else if(ib->dataType == LB_UINT16 ){ ib->dataClassID[i] =
    m_blockManager.setDataClass< Scalar3D<u16>, Scalar3DUpdater<u16> >(vc); }
650                                         else if(ib->dataType == LB_INT32  ){ ib->dataClassID[i] =
    m_blockManager.setDataClass< Scalar3D<s32>, Scalar3DUpdater<s32> >(vc); }
651                                         else if(ib->dataType == LB_UINT32 ){ ib->dataClassID[i] =
    m_blockManager.setDataClass< Scalar3D<u32>, Scalar3DUpdater<u32> >(vc); }
652                                         else if(ib->dataType == LB_INT64  ){ ib->dataClassID[i] =
    m_blockManager.setDataClass< Scalar3D<s64>, Scalar3DUpdater<s64> >(vc); }
653                                         else if(ib->dataType == LB_UINT64 ){ ib->dataClassID[i] =
    m_blockManager.setDataClass< Scalar3D<u64>, Scalar3DUpdater<u64> >(vc); }
654                                         #endif
655                                         m_blockManager.prepareForVCUpdate(ib->dataClassID[i],
     GetUniqueTag(), separateVCUpdate);
656                                         ib->separateVCUpdate = separateVCUpdate;
657                                 }
658                         }
659                         for(int i = 0; i < static_cast<int>(ib->kind); i++){
660                                 dataClassID[i] = ib->dataClassID[i];
661                         }
662                 }
663 /*
664                 int dataClassID = ib->dataClassID;
665                 // name に対応するブロックが作成されていない場合、作成 (初期化)
666                 if(dataClassID < 0){
667                         if(ib->kind == LB_CELLID)
668                         {
669                                 // CellID 向けブロック
670                                 //dataClassID = m_blockManager.setDataClass< Scalar3D<unsigned char>,
     Scalar3DUpdater<unsigned char> >(vc);
671                                 dataClassID = m_blockManager.setDataClass< Scalar3D<unsigned char> >(vc);
    // TODO Cell Updater
672                         }
673                         else
674                         {
675                                 // Scalar 向けブロック (Type 別)
676                                 if     ( ib->dataType == LB_FLOAT32) { dataClassID =
     m_blockManager.setDataClass< Scalar3D<f32>, Scalar3DUpdater<f32> >(vc); }
677                                 else if( ib->dataType == LB_FLOAT64) { dataClassID =
     m_blockManager.setDataClass< Scalar3D<f64>, Scalar3DUpdater<f64> >(vc); }
678                                 //else if( ib->dataType == LB_INT8   ) { dataClassID =
     m_blockManager.setDataClass< Scalar3D< s8>, Scalar3DUpdater< s8> >(vc); }
679                                 //else if( ib->dataType == LB_UINT8  ) { dataClassID =
     m_blockManager.setDataClass< Scalar3D< u8>, Scalar3DUpdater< u8> >(vc); }
680                                 //else if( ib->dataType == LB_INT16  ) { dataClassID =
     m_blockManager.setDataClass< Scalar3D<s16>, Scalar3DUpdater<s16> >(vc); }
681                                 //else if( ib->dataType == LB_UINT16 ) { dataClassID =
     m_blockManager.setDataClass< Scalar3D<u16>, Scalar3DUpdater<u16> >(vc); }
682                                 //else if( ib->dataType == LB_INT32  ) { dataClassID =
     m_blockManager.setDataClass< Scalar3D<s32>, Scalar3DUpdater<s32> >(vc); }
683                                 //else if( ib->dataType == LB_UINT32 ) { dataClassID =
     m_blockManager.setDataClass< Scalar3D<u32>, Scalar3DUpdater<u32> >(vc); }
```

```
684                                 //else if( ib->dataType == LB_INT64  ) { dataClassID =
    m_blockManager.setDataClass< Scalar3D<s64>, Scalar3DUpdater<s64> >(vc); }
685                                 //else if( ib->dataType == LB_UINT64 ) { dataClassID =
    m_blockManager.setDataClass< Scalar3D<u64>, Scalar3DUpdater<u64> >(vc); }
686                                 else{
687                                         Logger::Error("invalid DataType (%d)[%s:%d]\n", ib->dataType,
    __FILE__, __LINE__);
688                                         err = true;
689                                 }
690                                 m_blockManager.prepareForVCUpdate(dataClassID, GetUniqueTag(),
    separateVCUpdate);
691                                 ib->separateVCUpdate = separateVCUpdate;
692                         }
693
694                         ib->dataClassID = dataClassID;
695                 }
696 */
697                 if( ErrorUtil::reduceError(err) ){ return false; }
698
699                 return true;
700         }
```

5.1.3.3 Vec3d BCMFileIO::BCMFileLoader::GetGlobalOrigin ( ) const [inline]

領域全体の原点座標を返す.

戻り値

原点座標

BCMFileLoader.h の 112 行目に定義があります。

```
112 { return m_globalOrigin; }
```

5.1.3.4 Vec3d BCMFileIO::BCMFileLoader::GetGlobalRegion ( ) const [inline]

領域全体の長さを返す.

戻り値

領域の長さ

BCMFileLoader.h の 118 行目に定義があります。

```
118 { return m_globalRegion; }
```

5.1.3.5 BCMOctree* BCMFileIO::BCMFileLoader::GetOctree ( ) [inline]

読み込んだOctree を返す.

戻り値

Octree のポインタ

BCMFileLoader.h の 106 行目に定義があります。

```
106 { return m_octree; }
```

5.1.3.6 const IdxStep * BCMFileIO::BCMFileLoader::GetStep (const std::string & *name*) const

タイムステップ情報を取得

引数

in	*name*	系の名称

戻り値

name に対応したタイムステップのポインタ。系が存在しない場合NULL

BCMFileLoader.cpp の 818 行目に定義があります。

参照先 BCMFileIO::IdxBlock::find(), m_idxBlockList, BCMFileIO::IdxBlock::step.

```
819         {
820                 const IdxBlock* ib = IdxBlock::find(m_idxBlockList, name);
821
822                 if( ib == NULL ) return NULL;
823
824                 return &(ib->step);
825         }
```

5.1.3.7 bool BCMFileIO::BCMFileLoader::GetType (const std::string & *typeStr*, LB_DATA_TYPE & *retType*) [private]

LB_DATA_TYPE のデータタイプ取得

引数

in	*typeStr*	データタイプ
out	*retType*	LB_DATA_TYPE データタイプ

戻り値

true: 正常終了、false: 該当データタイプなし

BCMFileLoader.cpp の 893 行目に定義があります。

参照先 CompStr().

参照元 LoadIndexData().

```
893                                                                                 {
894                 bool status = false;
895                 const char *typeList[10] = {
896                         "Int8", "UInt8", "Int16", "UInt16", "Int32", "UInt32", "Int64", "UInt64", "Float32"
      , "Float64"
897                 };
898                 for(int i = 0; i < 10; i++){
899                         if(CompStr(typeStr, typeList[i]) == 0){
900                                 status = true;
901                                 retType = (LB_DATA_TYPE)(i);
902                                 break;
903                         }
904                 }
905                 return status;
906         }
```

5.1.3.8 int BCMFileIO::BCMFileLoader::GetUniqueTag ( ) [private]

ユニークなタグを取得

BCMFileLoader.cpp の 827 行目に定義があります。

参照元 CreateLeafBlock().

```
827                                     {
828                 static const int tagBase = 1000;
829                 static int       conter = 0;
830
831                 return tagBase + conter++;
832         }
```

5.1.3.9 const IdxUnit& BCMFileIO::BCMFileLoader::GetUnit ( ) const [inline]

単位系を取得

戻り値

単位系

BCMFileLoader.h の 131 行目に定義があります。

```
131 { return m_unit; }
```

5.1.3.10 bool BCMFileIO::BCMFileLoader::LoadAdditionalIndex (const std::string & *filepath*)

追加のインデックスファイルを読み込む

引数

in	*filepath*	インデックスファイルのファイル名 (実行ディレクトリからの相対パス)

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

cellid.bcm と data.bcm の 2 種類を読み込む場合に使用する。 読み込む順番は不問。 コンストラクタで読み込んだインデックスファイルとここで読み込むインデックスファイルの内容に相違がある場合、false を返す 内容の相違は、proc.bcm と tree.oct のヘッダ情報で判定

BCMFileLoader.cpp の 80 行目に定義があります。

参照先 BCMFileIO::FileSystemUtil::ConvertPath(), BCMFileIO::Logger::Error(), BCMFileIO::FileSystemUtil::GetDirectory(), LoadIndex(), LoadOctreeHeader(), m_idxBlockList, m_idxProcList, m_octree, BCMFileIO::OctHeader::numLeaf, BCMFileIO::ErrorUtil::reduceError().

```
81         {
82                 Vec3d org;
83                 Vec3d rgn;
84                 std::string octname;
85                 Vec3i blockSize;
86                 std::vector<IdxProc>  idxProcList;
87                 std::vector<IdxBlock> idxBlockList;
88
89                 if( ErrorUtil::reduceError( !LoadIndex(filepath, org, rgn,
     octname, blockSize, idxProcList, idxBlockList) ) ){
90                         Logger::Error("load index file error (%s) [%s:%d].\n", filepath.c_str(
     ), __FILE__, __LINE__);
91                         return false;
92                 }
93                 // Check error
94                 if( idxProcList.size() != m_idxProcList.size() ){
95                         Logger::Error("Process Info is invalid (%s) [%s:%d].\n", filepath.
```

```
      c_str(), __FILE__, __LINE__);
96                          return false;
97                  }
98                  // Check Octree
99                  {
100                         std::string dir = FileSystemUtil:: GetDirectory(
      FileSystemUtil::ConvertPath(filepath));
101                         std::string octFilepath = dir + octname;
102                         FILE *fp = NULL;
103                         if( (fp = fopen(octFilepath.c_str(), "rb")) == NULL ){
104                                 Logger::Error("Octree is invalid (%s) [%s:%d].\n", filepath.
      c_str(), __FILE__, __LINE__);
105                                 return false;
106                         }
107                         OctHeader hdr;
108                         bool isNeedSwap = false;
109                         if( !LoadOctreeHeader(fp, hdr, isNeedSwap) ){
110                                 Logger::Error("Octree is invalid (%s) [%s:%d].\n", filepath.
      c_str(), __FILE__, __LINE__);
111                                 fclose(fp);
112                                 return false;
113                         }
114                         fclose(fp);
115                         if( hdr.numLeaf != m_octree->getNumLeafNode() ){
116                                 Logger::Error("Octree is invalid (%s) [%s:%d].\n", filepath.
      c_str(), __FILE__, __LINE__);
117                                 return false;
118                         }
119                 }
120
121
122                 for(std::vector<IdxBlock>::iterator it = idxBlockList.begin(); it != idxBlockList.end(); ++
      it){
123                         m_idxBlockList.push_back(*it);
124                 }
125
126                 return true;
127         }
```

5.1.3.11 bool BCMFileIO::BCMFileLoader::LoadIndex (const std::string & *filename*, Vec3d & *globalOrigin*, Vec3d & *globalRegion*, std::string & *octreeFilename*, Vec3i & *blockSize*, std::vector< IdxProc > & *idxProcList*, std::vector< IdxBlock > & *idxBlockList*) [private]

Index ファイルを読み込む

引数

in	*filename*	インデックスファイル (cellid.bcm / data.bcm) の相対パス
out	*globalOrigin*	計算空間全体の起点座標
out	*globalRegion*	計算空間全体の領域サイズ
out	*octreeFilename*	Octree ファイルのファイル名
out	*blockSize*	ブロックサイズ
out	*idxProcList*	プロセス情報リスト
out	*idxBlockList*	ブロック情報リスト

戻り値

成功した場合 true, 失敗した場合 false

BCMFileLoader.cpp の 129 行目に定義があります。

参照先 CompStr(), BCMFileIO::FileSystemUtil::ConvertPath(), BCMFileIO::Logger::Error(), BCMFileIO::FileSystemUtil::FixDirectoryPath(), BCMFileIO::FileSystemUtil::GetDirectory(), BCMFileIO::IdxUnit::L0_scale, BCMFileIO::IdxUnit::length, LoadIndexCellID(), LoadIndexData(), LoadIndexProc(), m_unit, ReadVec3(), BCMFileIO::IdxBlock::rootDir, BCMFileIO::IdxUnit::V0_scale, BCMFileIO::IdxUnit::velocity (計 15 項目).

参照元 BCMFileLoader(), LoadAdditionalIndex().

```
131         {
```

```
132                 using namespace std;
133                 TextParser *tp = new TextParser;
134
135                 if( tp->read(filename) != TP_NO_ERROR ) {
136                         Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
137                         delete tp;
138                         return false;
139                 }
140
141                 tp->changeNode("/BCMTree");
142                 // Read Octree Filename
143                 if( tp->getValue("TreeFile", octreeFilename) != TP_NO_ERROR ){
144                         Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
145                         delete tp;
146                         return false;
147                 }
148
149                 // Read Proc Filename
150                 string procFilename;
151                 if( tp->getValue("ProcFile", procFilename) != TP_NO_ERROR ){
152                         Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
153                         delete tp;
154                         return false;
155                 }
156
157                 string dir = FileSystemUtil::GetDirectory(
    FileSystemUtil::ConvertPath(filename));
158                 string procFilepath = dir + procFilename;
159                 if( !LoadIndexProc(procFilepath, idxProcList) ){
160                         Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
161                         delete tp;
162                         return false;
163                 }
164
165                 // Read Domain
166                 tp->changeNode("/Domain");
167                 {
168                         bool hasOrigin = false;
169                         bool hasRegion = false;
170                         vector<string> lbls;
171                         tp->getLabels(lbls);
172                         for(vector<string>::iterator it = lbls.begin(); it != lbls.end(); ++it){
173                                 string valStr;
174                                 tp->getValue(*it, valStr);
175
176                                 if( CompStr(*it, "GlobalOrigin") == 0 ){
177                                         if( ReadVec3(tp, *it, globalOrigin) == TP_NO_ERROR ){
    hasOrigin = true; }
178                                         continue;
179                                 }
180                                 if( CompStr(*it, "GlobalRegion") == 0 ){
181                                         if( ReadVec3(tp, *it, globalRegion) == TP_NO_ERROR ){
    hasRegion = true; }
182                                         continue;
183                                 }
184                         }
185                         if( !hasOrigin || !hasRegion ){
186                                 Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
187                                 delete tp;
188                                 return false;
189                         }
190                 }
191
192                 tp->changeNode("/LeafBlock" );
193                 {
194                         vector<string> nodes;
195                         tp->getNodes(nodes, 1);
196                         for(vector<string>::iterator nit = nodes.begin(); nit != nodes.end(); ++nit){
197                                 // Load Data
198                                 if( CompStr(nit->substr(0, 4), "Data") == 0 ) {
199                                         tp->changeNode(*nit);
200                                         IdxBlock ib;
201                                         if( LoadIndexData(tp, &ib) ){
202                                                 ib.rootDir =
    FileSystemUtil::FixDirectoryPath(dir);
203                                                 idxBlockList.push_back(ib);
204                                         }
205                                         tp->changeNode("../");
206                                         continue;
207                                 }
208                                 // Load CellID
209                                 if( CompStr(nit->substr(0, 6), "CellID") == 0 ){
210                                         tp->changeNode(*nit);
211                                         IdxBlock ib;
212                                         if( LoadIndexCellID(tp, &ib) ){
213                                                 ib.rootDir =
    FileSystemUtil::FixDirectoryPath(dir);
```

```
214                                             idxBlockList.push_back(ib);
215                                     }
216                                     tp->changeNode("../");
217                                     continue;
218                             }
219                             // Load Unit
220                             if( CompStr(*nit, "Unit") == 0 ){
221                                     tp->changeNode(*nit);
222                                     vector<string> lbls;
223                                     tp->getLabels(lbls);
224                                     for(vector<string>::iterator it = lbls.begin(); it != lbls.end(); ++it){
225                                             string valStr;
226                                             tp->getValue(*it, valStr);
227
228                                             if( CompStr(*it, "Length")   == 0 ){ m_unit.length   = valStr; continue; }
229                                             if( CompStr(*it, "Velocity") == 0 ){ m_unit.velocity = valStr; continue; }
230                                             if( CompStr(*it, "L0") == 0 ){ m_unit.L0_scale = atof(valStr.c_str()); continue; }
231                                             if( CompStr(*it, "V0") == 0 ){ m_unit.V0_scale = atof(valStr.c_str()); continue; }
232                                     }
233                                     tp->changeNode("../");
234                             }
235                     }
236
237                     if( idxBlockList.size() == 0 ){
238                             Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
239                             delete tp;
240                             return false;
241                     }
242
243                     if( ReadVec3(tp, "size", blockSize) != TP_NO_ERROR ){
244                             Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
245                             delete tp;
246                             return false;
247                     }
248             }
249             delete tp;
250             return true;
251     }
```

5.1.3.12 bool BCMFileIO::BCMFileLoader::LoadIndexCellID (TextParser * *tp*, IdxBlock * *ib*) [private]

Index ファイルのリーフブロック情報を読み込む (cellId.bcm 用)

引数

in	*tp*	TextParser
out	*ib*	読み込んだブロック情報

戻り値

成功した場合 true, 失敗した場合 false

BCMFileLoader.cpp の 471 行目に定義があります。

参照先 BCMFileIO::IdxBlock::bitWidth, CompStr(), BCMFileIO::IdxBlock::dataDir, BCMFileIO::IdxBlock::dataType, BCMFileIO::Logger::Error(), BCMFileIO::IdxBlock::extension, BCMFileIO::FileSystemUtil::FixDirectoryPath(), BCMFileIO::IdxBlock::isGather, BCMFileIO::IdxBlock::kind, BCMFileIO::LB_CELLID, BCMFileIO::LB_UINT8, BCMFileIO::IdxBlock::name, BCMFileIO::IdxBlock::prefix, BCMFileIO::IdxBlock::vc.

参照元 LoadIndex().

```
472     {
473             using namespace std;
474             vector<string> lbls;
475             tp->getLabels(lbls);
476
477             bool hasName       = false;
478             bool hasBitWidth   = false;
```

```
479                 bool hasPrefix     = false;
480                 bool hasExtension  = false;
481                 bool hasGatherMode = false;
482 
483                 for(vector<string>::iterator it = lbls.begin(); it != lbls.end(); ++it){
484                         string valStr;
485                         tp->getValue(*it, valStr);
486 
487                         if( CompStr(*it, "name") == 0){
488                                 ib->name = valStr;
489                                 hasName = true;
490                                 continue;
491                         }
492 
493                         if( CompStr(*it, "BitWidth") == 0 ){
494                                 ib->bitWidth = atoi(valStr.c_str());
495                                 ib->kind     = LB_CELLID;
496                                 ib->dataType = LB_UINT8;
497                                 hasBitWidth  = true;
498                                 continue;
499                         }
500 
501                         if( CompStr(*it, "VirtualCellSize") == 0){
502                                 ib->vc = atoi(valStr.c_str());
503                                 continue;
504                         }
505 
506                         if( CompStr(*it, "Prefix") == 0 ){
507                                 ib->prefix = valStr;
508                                 hasPrefix  = true;
509                                 continue;
510                         }
511 
512                         if( CompStr(*it, "Extension") == 0 ){
513                                 ib->extension = valStr;
514                                 hasExtension = true;
515                                 continue;
516                         }
517 
518                         if( CompStr(*it, "DirectoryPath") == 0){
519                                 ib->dataDir = FileSystemUtil::FixDirectoryPath
    (valStr);
520                                 continue;
521                         }
522 
523                         if( CompStr(*it, "GatherMode") == 0 ){
524                                 if( CompStr(valStr, "distributed") == 0){
525                                         ib->isGather  = false;
526                                         hasGatherMode = true;
527                                 }else if( CompStr(valStr, "gathered") == 0){
528                                         ib->isGather  = true;
529                                         hasGatherMode = true;
530                                 }else{
531                                         Logger::Error("value (%s) of keyword [GatherMode] is
     invalid.\n", valStr.c_str());
532                                         return false;
533                                 }
534                                 continue;
535                         }
536                 }
537 
538                 if(!hasName || !hasBitWidth || !hasPrefix || !hasExtension || !hasGatherMode ){
539                         Logger::Error("Load Index File Error [%d:%s].\n", __FILE__, __LINE__);
540                         return false;
541                 }
542 
543                 return true;
544         }
```

5.1.3.13 bool BCMFileIO::BCMFileLoader::LoadIndexData (TextParser * *tp*, IdxBlock * *ib*) [private]

Index ファイルのリーフブロック情報を読み込む (data.bcm 用)

引数

in	*tp*	TextParser
out	*ib*	読み込んだブロック情報

戻り値

成功した場合 true, 失敗した場合 false

BCMFileLoader.cpp の 374 行目に定義があります。

参照先 BCMFileIO::IdxBlock::bitWidth, CompStr(), BCMFileIO::IdxBlock::dataClassID, BCMFileIO::IdxBlock::dataDir, BCMFileIO::IdxBlock::dataType, BCMFileIO::Logger::Error(), BCMFileIO::IdxBlock::extension, BCMFileIO::FileSystemUtil::FixDirectoryPath(), GetType(), BCMFileIO::IdxBlock::isStepSubDir, BCMFileIO::IdxBlock::kind, LoadIndexStep(), BCMFileIO::IdxBlock::name, BCMFileIO::IdxBlock::prefix, BCMFileIO::IdxBlock::step, BCMFileIO::IdxBlock::vc (計 16 項目).

参照元 LoadIndex().

```
375         {
376                 using namespace std;
377                 vector<string> lbls;
378                 tp->getLabels(lbls);
379 
380                 bool hasName          = false;
381                 bool hasType          = false;
382                 bool hasNumComponents = false;
383                 bool hasVC            = false;
384                 bool hasPrefix        = false;
385                 bool hasExtension     = false;
386 
387                 for(vector<string>::iterator it = lbls.begin(); it != lbls.end(); ++it){
388                         string valStr;
389                         tp->getValue(*it, valStr);
390 
391                         if( CompStr(*it, "name") == 0 ){
392                                 ib->name = valStr;
393                                 hasName = true;
394                                 continue;
395                         }
396 
397                         if( CompStr(*it, "type") == 0 ){
398                                 if( !GetType(valStr, ib->dataType) ){
399                                         Logger::Error("value (%s) of keyword [type] is invalid
    ", valStr.c_str());
400                                         return false;
401                                 }
402                                 hasType = true;
403                                 continue;
404                         }
405 
406                         if( CompStr(*it, "NumberOfComponents") == 0 ){
407                                 int val = atoi(valStr.c_str());
408                                 if( val != 1 && val != 3 && val != 4 && val != 6 && val != 9){
409                                         Logger::Error("value (%d) of keyword
     [NumberOfComponents] is invalid", val);
410                                         return false;
411                                 }
412                                 ib->kind = static_cast<LB_KIND>(val);
413                                 hasNumComponents = true;
414                                 continue;
415                         }
416 
417                         if( CompStr(*it, "VirtualCellSize") == 0 ){
418                                 ib->vc = atoi(valStr.c_str());
419                                 hasVC = true;
420                                 continue;
421                         }
422 
423                         if( CompStr(*it, "Prefix") == 0 ){
424                                 ib->prefix = valStr;
425                                 hasPrefix = true;
426                                 continue;
427                         }
428 
429                         if( CompStr(*it, "Extension") == 0 ){
430                                 ib->extension = valStr;
431                                 hasExtension = true;
432                                 continue;
433                         }
434 
435                         if( CompStr(*it, "DirectoryPath") == 0){
```

```
436                                 ib->dataDir = FileSystemUtil::FixDirectoryPath
    (valStr);
437                                 continue;
438                         }
439
440                         if( CompStr(*it, "StepSubDirectory") == 0){
441                                 if( CompStr(valStr, "true") == 0 ){
442                                         ib->isStepSubDir = true;
443                                 }else{
444                                         ib->isStepSubDir = false;
445                                 }
446                         }
447                 }
448
449                 if( !hasName || !hasType || !hasNumComponents || !hasVC || !hasPrefix || !hasExtension ){
450                         Logger::Error("Load Index File Error [%d:%s].\n", __FILE__, __LINE__);
451                         return false;
452                 }
453
454                 ib->dataClassID.clear();
455                 int bitWidthTable[10] = {
456                         8, 8, 16, 16, 32, 32, 64, 64, 32, 64
457                 };
458                 ib->bitWidth = bitWidthTable[(int)(ib->dataType)];
459
460                 if( tp->changeNode("Step") != TP_NO_ERROR ){
461                         return false;
462                 }else{
463                         bool err = !LoadIndexStep(tp, &ib->step);
464                         tp->changeNode("../");
465                         if( err ){ return false; }
466                 }
467
468                 return true;
469         }
```

5.1.3.14 bool BCMFileIO::BCMFileLoader::LoadIndexProc (const std::string & *filename*, std::vector< IdxProc > & *procList*) [private]

Index ファイルのプロセス情報を読み込む

引数

in	*filename*	プロセス情報ファイルのパス (proc.bcm)
out	*procList*	プロセス情報リスト

戻り値

成功した場合 true, 失敗した場合 false

BCMFileLoader.cpp の 253 行目に定義があります。

参照先 CompStr(), BCMFileIO::Logger::Error(), BCMFileIO::IdxProc::hostname, BCMFileIO::IdxProc::rangeMax, BCMFileIO::IdxProc::rangeMin, BCMFileIO::IdxProc::rank.

参照元 LoadIndex().

```
254         {
255                 using namespace std;
256                 TextParser *tp = new TextParser;
257
258                 if( tp->read(filepath) != TP_NO_ERROR ) {
259                         Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
260                         delete tp;
261                         return false;
262                 }
263
264                 int numProcs = 0;
265                 {
266                         tp->changeNode("/MPI");
267                         string valStr;
268                         if( tp->getValue("NumberOfRank", valStr) != TP_NO_ERROR ){
269                                 Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
270                                 delete tp;
```

```
271                             return false;
272                     }
273                     numProcs = atoi(valStr.c_str());
274             }
275
276             tp->changeNode("/process");
277             vector<string> nodes;
278             tp->getNodes(nodes, 1);
279             for(vector<string>::iterator nit = nodes.begin(); nit != nodes.end(); ++nit){
280                     if( CompStr(nit->substr(0, 4), "Rank") == 0 ){
281                             tp->changeNode(*nit);
282
283                             vector<string> lbls;
284                             tp->getLabels(lbls);
285
286                             IdxProc proc;
287                             for(vector<string>::iterator it = lbls.begin(); it != lbls.end(); ++it){
288                                     string valStr;
289                                     tp->getValue(*it, valStr);
290                                     if( CompStr(*it, "ID") == 0 ){
291                                             proc.rank = atoi(valStr.c_str());
292                                             continue;
293                                     }
294                                     if( CompStr(*it, "Hostname") == 0 ){
295                                             proc.hostname = valStr;
296                                             continue;
297                                     }
298
299                                     if( CompStr(*it, "BlockRange") == 0 ){
300                                             double range[3];
301                                             tp->splitRange(valStr, &range[0], &range[1], &range[2]);
302                                             proc.rangeMin = static_cast<unsigned int>(range[0]);
303                                             proc.rangeMax = static_cast<unsigned int>(range[1]);
304                                             continue;
305                                     }
306                             }
307                             procList.push_back(proc);
308
309                             tp->changeNode("../");
310                     }
311             }
312
313             if(procList.size() != numProcs){
314                     Logger::Error("[%s:%d]\n", __FILE__, __LINE__);
315                     delete tp;
316                     procList.clear();
317                     return false;
318             }
319
320             delete tp;
321             return true;
322     }
```

5.1.3.15 bool BCMFileIO::BCMFileLoader::LoadIndexStep (TextParser * *tp*, IdxStep * *step*) [private]

Index ファイルのステップ情報を読み込む

引数

in	*tp*	TextParser
out	*step*	読み込んだステップ情報

戻り値

成功した場合 true, 失敗した場合 false

BCMFileLoader.cpp の 324 行目に定義があります。

参照先 BCMFileIO::IdxStep::AddStep(), CompStr(), BCMFileIO::IdxStep::SetRange(), BCMFileIO::IdxStep::SubStep().

参照元 LoadIndexData().

```
325     {
```

```
326                 using namespace std;
327                 vector<string> lbls;
328                 tp->getLabels(lbls);
329
330                 bool hasBase = false;
331
332                 for(vector<string>::iterator it = lbls.begin(); it != lbls.end(); ++it){
333                         string valStr;
334                         tp->getValue(*it, valStr);
335
336                         if( CompStr(*it, "base") == 0 ){
337                                 double range[3];
338                                 if( tp->splitRange(valStr, &range[0], &range[1], &range[2]) != TP_NO_ERROR
    ){
339                                         return false;
340                                 }
341                                 step->SetRange((unsigned int)range[0], (unsigned int)range[1], (unsigned
    int)range[2]);
342                                 hasBase = true;
343                         }
344
345                         if( CompStr(*it, "add") == 0 ){
346                                 vector<double> list;
347                                 if( tp->splitList(valStr, list) != TP_NO_ERROR ){
348                                         return false;
349                                 }
350
351                                 for(vector<double>::iterator lit = list.begin(); lit != list.end(); ++lit){
352                                         step->AddStep((unsigned int)(*lit));
353                                 }
354                         }
355
356                         if( CompStr(*it, "sub") == 0 ){
357                                 vector<double> list;
358                                 if( tp->splitList(valStr, list) != TP_NO_ERROR ){
359                                         return false;
360                                 }
361
362                                 for(vector<double>::iterator lit = list.begin(); lit != list.end(); ++lit){
363                                         step->SubStep((unsigned int)(*lit));
364                                 }
365
366                         }
367                 }
368
369                 if( !hasBase ){ return false; }
370
371                 return true;
372         }
```

5.1.3.16 bool BCMFileIO::BCMFileLoader::LoadLeafBlock (int * *dataClassID*, const std::string & *name*, const unsigned int *vc*, const unsigned int *step* = `0`, const bool *separateVCUpdate* = `false`)

タイムステップ (step) におけるリーフブロックをファイルから読み込む.

引数

out	*dataClassID*	生成したブロックのデータクラスID (配列の先頭アドレス)
in	*name*	系の名称
in	*vc*	仮想セルサイズ
in	*step*	読み込むタイムステップ
in	*separateVC-Update*	仮想セルの同期方法フラグ. true の場合、3 軸方向別々に同期を行う

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

dataClassID の対象がGrid の場合、step は無視される． name に対応するブロックが生成されていない場合、ブロックを生成． separateVCUpdate は，読み込むデータクラスに対する初回呼び出しのみ有効 LoadLeafBlock()の前にCreateLeafBlock() で空のリーフブロックを作成していた場合，CreateLeafBlock()時に指定した separateVCUpdate が有効になる．

BCMFileLoader.cpp の 703 行目に定義があります。

参照先 BCMFileIO::LeafBlockLoader::CopyBufferToScalar3D(), CreateLeafBlock(), BCMFileIO::LeafBlockLoader::DecompCellIDData(), BCMFileIO::Logger::Error(), BCMFileIO::IdxBlock::find(), BCMFileIO::PartitionMapper::GetFDIDLists(), BCMFileIO::LB_CELLID, BCMFileIO::LeafBlockLoader::LoadCellID(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadData(), m_blockManager, m_comm, m_idxBlockList, m_leafBlockSize, m_pmapper, BCMFileIO::ErrorUtil::reduceError(), Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z (計 19 項目).

```
705         {
706                 using namespace std;
707
708                 const int myRank    = m_comm.Get_rank();
709
710                 bool err = false;
711
712                 IdxBlock* ib = IdxBlock::find(m_idxBlockList, name);
713
714                 if( ib == NULL ){
715                         Logger::Error("No such name as \"%s\" in loaded index.[%s:%d]\n", name
    .c_str(), __FILE__, __LINE__);
716                         err = true;
717                 }
718                 if( ErrorUtil::reduceError(err) ){ return false; }
719
720                 if( ErrorUtil::reduceError( !
    CreateLeafBlock(dataClassID, name, vc, separateVCUpdate) ) ){ return false; }
721
722                 // CellID データロード
723                 if(ib->kind == LB_CELLID)
724                 {
725                         LBHeader header;
726                         std::vector<CellIDCapsule> cidCapsules;
727
728                         if( ib->isGather ){
729                                 // ファイルから CellID を読み込む (GatherMode = Gathered)
730                                 err = !LeafBlockLoader::LoadCellID_Gather
    (ib->rootDir + ib->dataDir, ib, m_comm, m_pmapper, header, cidCapsules);
731                         }else{
732                                 // ファイルから CellID を読み込む (GatherMode = Distributed)
733                                 err = !LeafBlockLoader::LoadCellID(ib->rootDir +
     ib->dataDir, ib, m_comm, m_pmapper, header, cidCapsules);
734                         }
735                         // エラーチェック
736                         if( ErrorUtil::reduceError(err) ){
737                                 for(vector<CellIDCapsule>::iterator it = cidCapsules.begin(); it !=
    cidCapsules.end(); ++it){
738                                         delete [] it->data;
739                                 }
740                                 return false;
741                         }
742
743                         // MxN マッピング情報取得
744                         vector<PartitionMapper::FDIDList> fdidlists;
745                         m_pmapper->GetFDIDLists(myRank, fdidlists);
746
747                         int did = 0;
748                         int fid = 0;
749                         Vec3i bsz = Vec3i(m_leafBlockSize);
750                         // cidCapsules から逐次データを展開し BlockManager 配下の Block へデータをコピー
751                         for(vector<PartitionMapper::FDIDList>::iterator file = fdidlists.begin(); file !=
    fdidlists.end(); ++file){
752
753                                 // rle および bitVoxel の圧縮を展開
754                                 unsigned char* voxels =
    LeafBlockLoader::DecompCellIDData( header, cidCapsules[fid] );
755
756                                 // 展開した voxels からブロックごとにデータをコピー
757                                 for( vector<int>::iterator fdid = file->FDIDs.begin(); fdid != file->FDIDs.
    end(); ++fdid){
758                                         Vec3i ibsz( bsz.x + vc*2,     bsz.
    y + vc*2,     bsz.z + vc*2    );   // 内部ブロックサイズ (仮想セル込み)
759                                         Vec3i fbsz( bsz.x + ib->vc*2, bsz.
    y + ib->vc*2, bsz.z + ib->vc*2);   // ファイルブロックサイズ (仮想セル込み)
```

```
760                                     unsigned char* block = new unsigned char[ibsz.x * ibsz.y * ibsz.z];
        // データコピー用一時バッファを準備
761                                     memset(block, 0, sizeof(unsigned char) * ibsz.x * ibsz.y * ibsz.z);
        // 一時バッファの 0 クリア
762                                     unsigned char *pv = &voxels[ (fbsz.x * fbsz.y * fbsz.z) * (*fdid) ]
    ;
763
764                                     // ファイルから読み込んだ CellID を一時バッファにコピー （仮想セルサイズの不一致
への対応）
765                                     if( vc > ib->vc ){
766                                             unsigned int vcd = vc - ib->vc;
767                                             for(int z = 0; z < fbsz.z; z++){
768                                                     for(int y = 0; y < fbsz.y; y++){
769                                                             size_t ibloc = 0 + vcd + ( (y + vcd) + (z +
     vcd) * ibsz.y ) * ibsz.x;
770                                                             size_t fbloc = 0 +     + (  y        +  z
          * fbsz.y ) * fbsz.x;
771                                                             memcpy(&block[ibloc], &pv[fbloc], sizeof(
    unsigned char) * fbsz.x );
772                                                     }
773                                             }
774                                     }else{
775                                             unsigned int vcd = ib->vc - vc;
776                                             for(int z = 0; z < ibsz.z; z++){
777                                                     for(int y = 0; y < ibsz.y; y++){
778                                                             size_t ibloc = 0 +     + (  y        +  z
          * ibsz.y ) * ibsz.x;
779                                                             size_t fbloc = 0 + vcd + ( (y + vcd) + (z +
     vcd) * fbsz.y ) * fbsz.x;
780                                                             memcpy(&block[ibloc], &pv[fbloc], sizeof(
    unsigned char) * ibsz.x );
781                                                     }
782                                             }
783                                     }
784                                     // ブロックマネージャ配下のブロックに値をコピー
785                                     LeafBlockLoader::CopyBufferToScalar3D
    (m_blockManager, dataClassID[0], did, vc, block);
786                                     did++;
787                                     delete [] block;
788                                 }
789
790                                 delete [] voxels;
791                                 fid++;
792                             }
793                 }
794                 else
795                 {
796                             // ファイルからデータを読み込み、ブロックマネージャ配下のブロックに値をコピー
797                             if( ErrorUtil::reduceError(!
    LeafBlockLoader::LoadData( m_comm, ib,
    m_blockManager, m_pmapper, vc, step)) ){
798                                 return false;
799                             }
800                             // 現在の仮想セルサイズがファイルに記載されている仮想セルサイズよりも大きい場合、仮想セルの同期を行う
801                             if( vc > ib->vc ){
802                                     for(int i = 0; i < static_cast<int>(ib->kind); i++){
803                                     if( ib->separateVCUpdate ){
804                                             m_blockManager.updateVC_X(dataClassID[i]);
805                                             m_blockManager.updateVC_Y(dataClassID[i]);
806                                             m_blockManager.updateVC_Z(dataClassID[i]);
807                                     }
808                                     else{
809                                             m_blockManager.updateVC(dataClassID[i]);
810                                     }
811                                 }
812                             }
813                 }
814
815                 return true;
816         }
```

5.1.3.17 bool BCMFileIO::BCMFileLoader::LoadOctree (const std::string & *filename*, BoundaryConditionSetterBase * *bcsetter*) [private]

Octree ファイルを読み込む

引数

in	*filename*	Octree ファイルのファイル名
in	*bcsetter*	境界条件設定クラス

戻り値

成功した場合 true, 失敗した場合 false

BCMFileLoader.cpp の 547 行目に定義があります。

参照先 BCMFileIO::Logger::Error(), LoadOctreeFile(), m_blockManager, m_comm, m_globalOrigin, m_globalRegion, m_idxProcList, m_leafBlockSize, m_octree, m_pmapper, BCMFileIO::ErrorUtil::reduceError(), Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z.

参照元 BCMFileLoader().

```
548         {
549                 using namespace std;
550 
551                 int myRank   = m_comm.Get_rank();
552                 int numProcs = m_comm.Get_size();
553 
554                 bool load_octree_only_master = false;
555 #ifdef OCTREE_LOAD_ONLY_MASTER
556                 load_octree_only_master = true;
557 #endif
558 
559                 OctHeader header;
560                 vector<Pedigree> pedigrees;
561 
562                 if( load_octree_only_master )
563                 {
564                         unsigned char loadError = 0;
565                         if( myRank == 0 ){
566                                 if( !LoadOctreeFile(filename, header, pedigrees) ) loadError
    = 1;
567                         }
568 
569                         m_comm.Bcast(&loadError, 1, MPI::CHAR, 0);
570                         if( loadError == 1) return false;
571                         m_comm.Bcast(&header, sizeof(OctHeader), MPI::CHAR, 0);
572 
573                         if( myRank != 0 ){
574                                 pedigrees.resize(header.numLeaf);
575                         }
576                         m_comm.Bcast(&pedigrees[0], sizeof(Pedigree) * pedigrees.size(), MPI::CHAR, 0
    );
577                 }
578                 else
579                 {
580                         if( ErrorUtil::reduceError( !
    LoadOctreeFile(filename, header, pedigrees)) ) return false;
581                 }
582 
583 
584                 Vec3d rootRegion( header.rgn[0] / static_cast<double>(header.rootDims[0]),
585                                   header.rgn[1] / static_cast<double>(header.rootDims[1]),
586                                                   header.rgn[2] / static_cast<double>(header.rootDims[2]) )
    ;
587 
588                 if( fabs(rootRegion.x - rootRegion.y) >= 1.0e-10 || fabs(rootRegion.x - rootRegion.z) >= 1.
    0e-10 ) {
589                         Logger::Error("%lf %lf %lf\n", rootRegion.x, rootRegion.y, rootRegion.
    z);
590                         Logger::Error("RootGrid is not regular hexahedron. [%s:%d]\n",
    __FILE__, __LINE__);
591                         return false;
592                 }
593 
594                 RootGrid *rootGrid = new RootGrid(header.rootDims[0], header.rootDims[1], header.rootDims[2
    ]);
595                 m_octree  = new BCMOctree(rootGrid, pedigrees);
596                 m_pmapper = new PartitionMapper(m_idxProcList.size(), numProcs,
    header.numLeaf);
597 
598                 // Make and register Block
599                 Partition part(numProcs, header.numLeaf);
600                 BlockFactory factory(m_octree, &part, bcsetter, Vec3d(header.org), rootRegion.
    x, m_leafBlockSize);
601 
```

```
602                 vector<Node*>& leafNodeArray = m_octree->getLeafNodeArray();
603                 for(int id = part.getStart(myRank); id < part.getEnd(myRank); id++){
604                         Node* node   = leafNodeArray[id];
605                         Block* block = factory.makeBlock(node);
606                         m_blockManager.registerBlock(block);
607                 }
608
609                 m_blockManager.endRegisterBlock();
610
611                 m_globalOrigin = Vec3d(header.org);
612                 m_globalRegion = Vec3d(header.rgn);
613
614                 return true;
615         }
```

5.1.3.18 bool BCMFileIO::BCMFileLoader::LoadOctreeFile (const std::string & *filename*, OctHeader & *header*, std::vector< Pedigree > & *pedigrees*) [private]

Octree ファイルより Pedigree リストを取得する

引数

in	*filename*	ファイル名
out	*header*	OctTree ファイルヘッダ
out	*pedigrees*	Pedigree リスト

戻り値

true: 正常終了、false: ファイル読み込みエラー

BCMFileLoader.cpp の 933 行目に定義があります。

参照先 BCMFileIO::BSwap64(), BCMFileIO::Logger::Error(), LoadOctreeHeader(), BCMFileIO::OctHeader::numLeaf.

参照元 LoadOctree().

```
934         {
935                 using namespace std;
936                 FILE *fp = NULL;
937                 if( (fp = fopen(filename.c_str(), "rb")) == NULL ){
938                         Logger::Error("open file error(%s) [%s:%d].\n", filename.c_str(),
    __FILE__, __LINE__);
939                         return false;
940                 }
941
942                 bool isNeedSwap = false;
943                 if( !LoadOctreeHeader(fp, header, isNeedSwap) ){
944                         Logger::Error("Load Header error(%s) [%s:%d].\n", filename.c_str(),
    __FILE__, __LINE__);
945                         fclose(fp);
946                         return false;
947                 }
948
949                 pedigrees.resize(header.numLeaf);
950                 fread(&pedigrees[0], sizeof(Pedigree), header.numLeaf, fp);
951
952                 fclose(fp);
953
954                 if( isNeedSwap ){
955                         for(vector<Pedigree>::iterator it = pedigrees.begin(); it != pedigrees.end(); ++it)
    {
956                                 BSwap64(&(*it));
957                         }
958                 }
959                 return true;
960         }
```

5.1.3.19 bool BCMFileIO::BCMFileLoader::LoadOctreeHeader (FILE * *fp,* OctHeader & *header,* bool & *isNeedSwap*) [private]

Octree ヘッダの取得

引数

in	*fp*	ファイルポインタ
out	*header*	OctTree ファイルヘッダ
out	*isNeedSwap*	true: BSwap 必要、false: BSwap 不要

戻り値

true: 正常終了、false: 読み込みエラー

BCMFileLoader.cpp の 908 行目に定義があります。

参照先 BCMFileIO::BSwap32(), BCMFileIO::BSwap64(), BCMFileIO::OctHeader::identifier, BCMFileIO::OctHeader::maxLevel, BCMFileIO::OctHeader::numLeaf, OCTREE_FILE_IDENTIFIER, BCMFileIO::OctHeader::org, BCMFileIO::OctHeader::rgn, BCMFileIO::OctHeader::rootDims.

参照元 LoadAdditionalIndex(), LoadOctreeFile().

```
909         {
910                 isNeedSwap = false;
911                 fread(&header, sizeof(header), 1, fp);
912
913                 if( header.identifier != OCTREE_FILE_IDENTIFIER ){
914                         BSwap32(&header.identifier);
915
916                         if( header.identifier != OCTREE_FILE_IDENTIFIER ){
917                                 return false;
918                         }
919
920                         isNeedSwap = true;
921                         for(int i = 0; i < 3; i++){
922                                 BSwap64(&header.org[i]);
923                                 BSwap64(&header.rgn[i]);
924                                 BSwap32(&header.rootDims[i]);
925                         }
926                         BSwap32(&header.maxLevel);
927                         BSwap64(&header.numLeaf);
928                 }
929
930                 return true;
931         }
```

5.1.3.20 void BCMFileIO::BCMFileLoader::PrintIdxInformation ( ) [private]

読み込んだインデックスファイルの内容を stdout に出力

5.1.3.21 void BCMFileIO::BCMFileLoader::PrintOctreeInformation ( ) [private]

読み込んだOctree ファイルの内容を stdout に出力

5.1.3.22 int BCMFileIO::BCMFileLoader::ReadVec3 (TextParser * *tp*, const std::string & *label*, Vec3d & *v*) [private]

Vec3d 型のデータ読み込み

引数

in	*tp*	テキストパーサ
in	*label*	ラベル
out	*v*	Vec3d

戻り値

== TP_NO_ERROR: 正常終了、!= TP_NO_ERROR: エラー終了

BCMFileLoader.cpp の 846 行目に定義があります。

参照先 Vec3class::Vec3< T >::x.

参照元 LoadIndex().

```
847         {
848                 using namespace std;
849
850                 if(!tp){ return TP_ERROR; }
851
852                 int err = TP_NO_ERROR;
853                 string valStr;
854                 vector<string> vec_valStr;
855                 if( (err = tp->getValue(label, valStr)) != TP_NO_ERROR){ return err; }
856                 tp->splitVector(valStr, vec_valStr);
857
858                 Vec3d ret_v;
859                 ret_v.x = tp->convertDouble(vec_valStr[0], &err);
860                 ret_v.y = tp->convertDouble(vec_valStr[1], &err);
861                 ret_v.z = tp->convertDouble(vec_valStr[2], &err);
862
863                 v = ret_v;
864
865                 return TP_NO_ERROR;
866         }
```

5.1.3.23 int BCMFileIO::BCMFileLoader::ReadVec3 (TextParser * *tp*, const std::string & *label*, Vec3i & *v*) [private]

Vec3i 型のデータ読み込み

引数

in	*tp*	テキストパーサ
in	*label*	ラベル
out	*v*	Vec3i

戻り値

== TP_NO_ERROR: 正常終了、!= TP_NO_ERROR: エラー終了

BCMFileLoader.cpp の 868 行目に定義があります。

参照先 Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z.

```
869         {
870                 using namespace std;
871
872                 if(!tp){ return TP_ERROR; }
873
874                 int err = TP_NO_ERROR;
875                 string valStr;
876                 vector<string> vec_valStr;
877
878                 if( (err = tp->getValue(label, valStr)) != TP_NO_ERROR){ return err; }
879                 tp->splitVector(valStr, vec_valStr);
880
881                 Vec3i ret_v;
882                 ret_v.x = tp->convertInt(vec_valStr[0], &err);
883                 ret_v.y = tp->convertInt(vec_valStr[1], &err);
```

```
884                 ret_v.z = tp->convertInt(vec_valStr[2], &err);
885
886                 v.x = ret_v.x;
887                 v.y = ret_v.y;
888                 v.z = ret_v.z;
889
890                 return TP_NO_ERROR;
891         }
```

5.1.4 メンバ詳解

5.1.4.1 BlockManager& BCMFileIO::BCMFileLoader::m_blockManager [private]

ブロックマネージャ

BCMFileLoader.h の 243 行目に定義があります。

参照元 CreateLeafBlock(), LoadLeafBlock(), LoadOctree().

5.1.4.2 const MPI::Intracomm& BCMFileIO::BCMFileLoader::m_comm [private]

MPI コミュニケータ

BCMFileLoader.h の 244 行目に定義があります。

参照元 LoadLeafBlock(), LoadOctree().

5.1.4.3 Vec3d BCMFileIO::BCMFileLoader::m_globalOrigin [private]

領域全体の原点座標

BCMFileLoader.h の 251 行目に定義があります。

参照元 BCMFileLoader(), LoadOctree().

5.1.4.4 Vec3d BCMFileIO::BCMFileLoader::m_globalRegion [private]

領域全体の長さ

BCMFileLoader.h の 252 行目に定義があります。

参照元 BCMFileLoader(), LoadOctree().

5.1.4.5 std::vector<IdxBlock> BCMFileIO::BCMFileLoader::m_idxBlockList [private]

ブロック情報リスト

BCMFileLoader.h の 248 行目に定義があります。

参照元 BCMFileLoader(), CreateLeafBlock(), GetStep(), LoadAdditionalIndex(), LoadLeafBlock().

5.1.4.6 std::vector<IdxProc> BCMFileIO::BCMFileLoader::m_idxProcList [private]

プロセス情報リスト

BCMFileLoader.h の 247 行目に定義があります。

参照元 BCMFileLoader(), LoadAdditionalIndex(), LoadOctree().

5.1.4.7 Vec3i BCMFileIO::BCMFileLoader::m_leafBlockSize [private]

リーフブロックサイズ

BCMFileLoader.h の 246 行目に定義があります。

参照元 BCMFileLoader(), LoadLeafBlock(), LoadOctree().

5.1.4.8 BCMOctree* BCMFileIO::BCMFileLoader::m_octree [private]

Octree.

BCMFileLoader.h の 254 行目に定義があります。

参照元 LoadAdditionalIndex(), LoadOctree(), ~BCMFileLoader().

5.1.4.9 PartitionMapper* BCMFileIO::BCMFileLoader::m_pmapper [private]

MxN データマッパ

BCMFileLoader.h の 256 行目に定義があります。

参照元 LoadLeafBlock(), LoadOctree(), ~BCMFileLoader().

5.1.4.10 IdxUnit BCMFileIO::BCMFileLoader::m_unit [private]

単位系

BCMFileLoader.h の 249 行目に定義があります。

参照元 LoadIndex().

このクラス詳解は次のファイルから抽出されました:

- BCMFileLoader.h
- BCMFileLoader.cpp

5.2 BCMFileIO::BCMFileSaver クラス

BCM ファイルを出力するクラス

`#include <BCMFileSaver.h>`

BCMFileIO::BCMFileSaver 連携図

公開メンバ関数

- BCMFileSaver (const Vec3d &globalOrigin, const Vec3d &globalRegion, const BCMOctree ∗octree, const std::string dir=std::string(""))
- ∼BCMFileSaver ()
  デストラクタ
- bool RegisterCellIDInformation (const int dataClassID, const unsigned int bitWidth, const short vc, const std::string &name, const std::string &prefix, const std::string &extension, const std::string &dataDir=std::string("./"), const bool gatherMode=true)
- bool RegisterDataInformation (const int ∗dataClassID, const LB_KIND kind, const LB_DATA_TYPE dataType, const short vc, const std::string &name, const std::string &prefix, const std::string &extension, const IdxStep &step, const std::string &dataDir=std::string("./"), const bool stepSubDir=false)
- bool SetUnit (const IdxUnit &unit)
- bool Save ()
- bool SaveLeafBlock (const char ∗name, unsigned int step=0)

非公開メンバ関数

- bool SaveIndex (const std::string &octName, const int numLeaf)
- bool SaveIndexProc (const std::string &filepath, const Partition &part)
- bool SaveIndexCellID (const std::string &procName, const std::string &octName)
- bool SaveIndexData (const std::string &procName, const std::string &octName)
- bool SaveOctree (const std::string &filepath, const BCMOctree ∗octree)
- unsigned char ∗ GetCellIDBlock (const IdxBlock ∗ib, BlockManager &blockManager)

非公開変数類

- BlockManager & m_blockManager
  ブロックマネージャ
- const MPI::Intracomm & m_comm
  MPI **コミュニケータ**
- const BCMOctree ∗ m_octree
  BCMOctree.
- const Vec3d m_globalOrigin
  計算空間の起点座標
- const Vec3d m_globalRegion
  計算空間全体の領域サイズ
- IdxUnit m_unit
  単位系
- std::string m_targetDir
  ファイル出力ターゲットディレクトリ名
- std::vector< IdxBlock > m_idxBlockList
  登録されたブロック情報リスト

5.2.1 詳解

BCM ファイルを出力するクラス

BCMFileSaver.h の 37 行目に定義があります。

5.2.2 構築子と解体子

5.2.2.1 BCMFileIO::BCMFileSaver::BCMFileSaver (const Vec3d & *globalOrigin,* const Vec3d & *globalRegion,* const BCMOctree * *octree,* const std::string *dir* = `std::string("")`)

コンストラクタ

引数

in	*globalOrigin*	計算空間全体の起点座標
in	*globalRegion*	計算空間全体の領域サイズ
in	*octree*	出力Octree
in	*dir*	ファイル出力先ディレクトリ (省略した場合、カレントディレクトリ)

BCMFileSaver.cpp の 43 行目に定義があります。

参照先 BCMFileIO::FileSystemUtil::FixDirectoryPath(), BCMFileIO::IdxUnit::L0_scale, BCMFileIO::IdxUnit::length, m_targetDir, m_unit, BCMFileIO::IdxUnit::V0_scale, BCMFileIO::IdxUnit::velocity.

```
44          : m_blockManager(BlockManager::getInstance()), m_comm(
    m_blockManager.getCommunicator()),
45            m_octree(octree), m_globalOrigin(globalOrigin),
    m_globalRegion(globalRegion)
46        {
47                m_targetDir = FileSystemUtil::FixDirectoryPath(
    dir);
48
49                m_unit.length   = std::string("NonDimensional");
50                m_unit.L0_scale = 1.0;
51                m_unit.velocity = std::string("NonDimensional");
52                m_unit.V0_scale = 1.0;
53        }
```

5.2.2.2 BCMFileIO::BCMFileSaver::∼BCMFileSaver ( )

デストラクタ

BCMFileSaver.cpp の 55 行目に定義があります。

```
56        {
57
58        }
```

5.2.3 関数詳解

5.2.3.1 unsigned char * BCMFileIO::BCMFileSaver::GetCellIDBlock (const IdxBlock * *ib*, BlockManager & *blockManager*) [private]

CellID ブロックを取得

引数

in	*ib*	インデックスブロック
in	*blockManager*	BlockManager

戻り値

CellID ブロックの先頭アドレス

BCMFileSaver.cpp の 570 行目に定義があります。

参照先 BCMFileIO::IdxBlock::dataClassID, BCMFileIO::IdxBlock::vc, Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z.

参照元 SaveLeafBlock().

```
571         {
572                 int vc = ib->vc;
573
574                 const Vec3i size = blockManager.getSize();
575                 const size_t numBlock = blockManager.getNumBlock();
576                 const size_t tsz = (size.x + vc*2) * (size.y + vc*2) * (size.z + vc*2) * numBlock;
577
578                 unsigned char* grids = new unsigned char[tsz];
579
580                 for(int id = 0; id < numBlock; ++id){
581                         BlockBase* block = blockManager.getBlock(id);
582
583                         Scalar3D<unsigned char>* mesh = dynamic_cast< Scalar3D<unsigned char>* >(block->
    getDataClass(ib->dataClassID[0]));
584                         unsigned char* data = mesh->getData();
585                         Index3DS idx = mesh->getIndex();
586
587                         Vec3i sz( size.x + vc*2, size.y + vc*2, size.z + vc*2);
588                         for(int z = 0; z < sz.z; z++){
589                                 for(int y = 0; y < sz.y; y++){
590                                         for(int x = 0; x < sz.x; x++){
591                                                 size_t loc = x + (y + (z + id * sz.z) * sz.y ) * sz.x;
592                                                 grids[loc] = data[ idx( x-vc, y-vc, z-vc ) ];
593                                         }
594                                 }
595                         }
596                 }
597                 return grids;
598         }
```

5.2.3.2 bool BCMFileIO::BCMFileSaver::RegisterCellIDInformation (const int *dataClassID*, const unsigned int *bitWidth*, const short *vc*, const std::string & *name*, const std::string & *prefix*, const std::string & *extension*, const std::string & *dataDir* = `std::string("./")`, const bool *gatherMode* = `true`)

出力対象のリーフブロック情報を登録 (CellID 用)

引数

in	*dataClassID*	データクラスID
in	*bitWidth*	CellID の表現ビット幅
in	*vc*	仮想セルサイズ
in	*name*	系の名称
in	*prefix*	リーフブロックファイルのPrefix
in	*extension*	リーフブロックファイルの拡張子
in	*dataDir*	リーフブロックファイルの出力ディレクトリを指定 (コンストラクタで指定した出力ディレクトリからの相対パス)
in	*gatherMode*	集約モード (true の場合、Rank 0 に集約)

戻り値

成功した場合 true, 失敗した場合 false

BCMFileSaver.cpp の 60 行目に定義があります。

参照先 BCMFileIO::IdxBlock::bitWidth, BCMFileIO::IdxBlock::dataClassID, BCMFileIO::IdxBlock::dataDir, BCMFileIO::IdxBlock::dataType, BCMFileIO::Logger::Error(), BCMFileIO::IdxBlock::extension, BCMFileIO::IdxBlock::find(), BCMFileIO::FileSystemUtil::FixDirectoryPath(), BCMFileIO::IdxBlock::isGather, BCMFileIO::IdxBlock::kind, BCMFileIO::LB_CELLID, BCMFileIO::LB_UINT8, m_idxBlockList, m_targetDir, BCMFileIO::IdxBlock::name, BCMFileIO::IdxBlock::prefix, BCMFileIO::IdxBlock::rootDir, BCMFileIO::IdxBlock::vc (計 18 項目).

```
69         {
70                 if(dataClassID < 0){ return false; }
71
72                 if( IdxBlock::find(m_idxBlockList, dataClassID) != NULL ){
73                         Logger::Error("dataClassID(%s) is already registerd [%s:%d]\n",
    dataClassID, __FILE__, __LINE__);
74                         return false;
75                 }
```

```
76
77                  IdxBlock ib;
78                  ib.dataClassID.resize(1);
79                  ib.dataClassID[0] = dataClassID;
80                  ib.rootDir     = m_targetDir;
81                  ib.dataDir     = FileSystemUtil::FixDirectoryPath(dataDir);
82                  ib.kind        = LB_CELLID;
83                  ib.dataType    = LB_UINT8;
84                  ib.bitWidth    = bitWidth;
85                  ib.vc          = vc;
86                  ib.name        = name;
87                  ib.prefix      = prefix;
88                  ib.extension   = extension;
89                  ib.isGather    = gather;
90
91                  m_idxBlockList.push_back(ib);
92
93                  return true;
94          }
```

5.2.3.3 bool BCMFileIO::BCMFileSaver::RegisterDataInformation (const int * *dataClassID*, const LB_KIND *kind*, const LB_DATA_TYPE *dataType*, const short *vc*, const std::string & *name*, const std::string & *prefix*, const std::string & *extension*, const IdxStep & *step*, const std::string & *dataDir* = `std::string("./")`, const bool *stepSubDir* = `false`)

出力対象のリーフブロック情報を登録 (Data 用)

引数

in	*dataClassID*	データクラスID (配列)
in	*kind*	データの種類
in	*dataType*	データの型
in	*vc*	仮想セルサイズ
in	*name*	系の名称
in	*prefix*	リーフブロックファイルのPrefix
in	*extension*	リーフブロックファイルの拡張子
in	*step*	タイムステップ情報
in	*dataDir*	リーフブロックファイルの出力ディレクトリを指定 (コンストラクタで指定した出力ディレクトリからの相対パス)
in	*stepSubDir*	タイムステップごとの出力ディレクトリフラグ (true の場合、タイムステップごとのディレクトリを作成)

戻り値

成功した場合 true, 失敗した場合 false

BCMFileSaver.cpp の 96 行目に定義があります。

参照先 BCMFileIO::IdxBlock::bitWidth, BCMFileIO::IdxBlock::dataClassID, BCMFileIO::IdxBlock::dataDir, BCMFileIO::IdxBlock::dataType, BCMFileIO::Logger::Error(), BCMFileIO::IdxBlock::extension, BCMFileIO::IdxBlock::find(), BCMFileIO::FileSystemUtil::FixDirectoryPath(), BCMFileIO::IdxBlock::isStepSubDir, BCMFileIO::IdxBlock::kind, m_idxBlockList, m_targetDir, BCMFileIO::IdxBlock::name, BCMFileIO::IdxBlock::prefix, BCMFileIO::IdxBlock::rootDir, BCMFileIO::IdxBlock::step, BCMFileIO::IdxBlock::vc (計 17 項目).

```
106         {
107                 if(!dataClassID){ return false; }
108                 for(int i = 0; i < static_cast<int>(kind); i++){
109                         if(dataClassID[i] < 0){
110                                 Logger::Error("dataClassID(%d) is invalid) [%s:%d]\n",
    dataClassID[i], __FILE__, __LINE__);
111                                 return false;
112                         }
113
114                         if( IdxBlock::find(m_idxBlockList, dataClassID[i]) !=
    NULL ){
115                                 Logger::Error("dataClassID(%d) is already registerd [%s:%d]\n"
```

```
      , dataClassID[i], __FILE__, __LINE__);
116                             return false;
117                     }
118             }
119
120
121             unsigned int bitWidthTable[10] = {
122                     8, 8, 16, 16, 32, 32, 64, 64, 32, 64
123             };
124
125
126             IdxBlock ib;
127
128             ib.dataClassID.resize(static_cast<int>(kind));
129             for(int i = 0; i < static_cast<int>(kind); i++){
130                     ib.dataClassID[i]  = dataClassID[i];
131             }
132
133             ib.rootDir      = m_targetDir;
134             ib.dataDir      = FileSystemUtil::FixDirectoryPath(dataDir)
      ;
135             ib.kind         = kind;
136             ib.dataType     = dataType;
137             ib.bitWidth     = bitWidthTable[(int)dataType];
138             ib.vc           = vc;
139             ib.name         = name;
140             ib.prefix       = prefix;
141             ib.extension    = extension;
142             ib.isStepSubDir = stepSubDir;
143             ib.step         = step;
144
145             m_idxBlockList.push_back(ib);
146
147             return true;
148         }
```

5.2.3.4 bool BCMFileIO::BCMFileSaver::Save ( )

ファイル出力を実行

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

ここで出力される情報は、インデックスファイルとOctree。 RegisterCellIDInformation()でCellID が登録されていない場合、cellid.bcm は出力されない RegisterDataInformation()でData が登録されていない場合、data.bcm は出力されない RegisterDataInformation()で複数のData を登録している場合、data.bcm にまとめて記載

BCMFileSaver.cpp の 157 行目に定義があります。

参照先 BCMFileIO::FileSystemUtil::CreateDirectory(), BCMFileIO::Logger::Error(), m_comm, m_octree, m_targetDir, BCMFileIO::ErrorUtil::reduceError(), SaveIndex(), SaveOctree().

```
158         {
159                 using namespace std;
160
161                 string octFilename("tree.oct");
162                 string octFilepath = m_targetDir + octFilename;
163                 //if( m_comm.Get_rank() == 0) Logger::Info("Output Files are : IDX[%s], OCT[%s]\n",
       idxFilepath.c_str(), octFilepath.c_str());
164
165                 bool err = false;
166
167                 if( m_comm.Get_rank() == 0 ){
168                         err = !FileSystemUtil::CreateDirectory(
      m_targetDir, m_targetDir.find("/") == 0 ? true : false);
169                 }else{
170                         err = false;
171                 }
```

```
172                 if( ErrorUtil::reduceError(err) ){
173                         return false;
174                 }
175
176                 err = ErrorUtil::reduceError( !SaveIndex(octFilename,
    m_octree->getNumLeafNode()) );
177                 if( err ){
178                         Logger::Error("faild to save index file. [%s:%d]\n", __FILE__,
    __LINE__);
179                         return false;
180                 }
181
182                 err = ErrorUtil::reduceError( !SaveOctree(octFilepath,
    m_octree) );
183                 if( err ){
184                         Logger::Error("faild to save octree file. [%s:%d]\n", __FILE__,
    __LINE__);
185                         return false;
186                 }
187
188                 return true;
189         }
```

5.2.3.5 bool BCMFileIO::BCMFileSaver::SaveIndex (const std::string & *octName*, const int *numLeaf*) [private]

インデックスファイルを出力

引数

in	*octName*	Octree ファイル名
in	*numLeaf*	総リーフブロック数

戻り値

成功した場合 true, 失敗した場合 false

BCMFileSaver.cpp の 494 行目に定義があります。

参照先 BCMFileIO::Logger::Error(), m_comm, m_targetDir, BCMFileIO::ErrorUtil::reduceError(), SaveIndexCellID(), SaveIndexData(), SaveIndexProc().

参照元 Save().

```
495         {
496                 using namespace std;
497
498                 if( numLeaf < m_comm.Get_size() ){
499                         Logger::Error("less number of leafs than number of procs. [%s:%d]\n",
    __FILE__, __LINE__);
500                         return false;
501                 }
502
503                 ostringstream os;
504                 os.setf(ios::scientific);
505                 os.precision(6);
506
507                 Partition part(m_comm.Get_size(), numLeaf);
508
509                 std::string procName("proc.bcm");
510                 std::string procPath = m_targetDir + procName;
511                 if( ErrorUtil::reduceError(!SaveIndexProc(procPath, part
    )) )       { Logger::Error("%s:%s:%d\n", __func__, __FILE__, __LINE__); return false; }
512                 if( ErrorUtil::reduceError(!
    SaveIndexCellID(procName, octName)) ) { Logger::Error("%s:%s:%d\n", __func__,
    __FILE__, __LINE__); return false; }
513                 if( ErrorUtil::reduceError(!SaveIndexData(procName,
    octName)) )    { Logger::Error("%s:%s:%d\n", __func__, __FILE__, __LINE__); return false; }
514
515                 return true;
516         }
```

5.2.3.6 bool BCMFileIO::BCMFileSaver::SaveIndexCellID (const std::string & *procName,* const std::string & *octName*) [private]

CellID 情報ファイルを出力

引数

in	*procName*	プロセス情報ファイルの名前
in	*octName*	Octree ファイルの名前

戻り値

成功した場合 true, 失敗した場合 false

BCMFileSaver.cpp の 337 行目に定義があります。

参照先 BCMFileIO::Logger::Error(), BCMFileIO::IdxUnit::L0_scale, BCMFileIO::LB_CELLID, BCMFileIO::IdxUnit::length, m_blockManager, m_comm, m_globalOrigin, m_globalRegion, m_idxBlockList, m_targetDir, m_unit.

参照元 SaveIndex().

```
338         {
339                 using namespace std;
340                 if( m_comm.Get_rank() != 0 ){ return true; }
341
342                 // search CellID IdxBlock
343                 IdxBlock *ib = NULL;
344                 for(vector<IdxBlock>::iterator it = m_idxBlockList.begin(); it !=
    m_idxBlockList.end(); ++it){
345                         if( it->kind == LB_CELLID ){
346                                 ib = &(*it);
347                                 break;
348                         }
349                 }
350                 if( !ib ) {
351                         return true;
352                 }
353
354                 Vec3i leafSize = m_blockManager.getSize();
355
356                 ostringstream os;
357                 os.setf(ios::scientific);
358                 os.precision(6);
359
360                 // write CellID Information
361                 os << "Domain { " << endl;
362                 os << "  GlobalOrigin = " << m_globalOrigin << endl;
363                 os << "  GlobalRegion = " << m_globalRegion << endl;
364                 os << "}" << endl;
365                 os << endl;
366                 os << "BCMTree {" << endl;
367                 os << "  TreeFile  = \"" << octName  << "\"" << endl;
368                 os << "  ProcFile  = \"" << procName << "\"" << endl;
369                 os << "}" << endl;
370                 os << endl;
371                 os << "LeafBlock {" << endl;
372                 os << "  size = " << leafSize << endl << endl;
373                 os << "  Unit {" << endl;
374                 os << "    Length = \"" << m_unit.length << "\"" << endl;
375                 os << "    L0     = " << m_unit.L0_scale << endl;
376                 os << "  }" << endl << endl;
377                 os << "  CellID { " << endl;
378                 os << "    name            = \"" << ib->name      << "\"" << endl;
379                 os << "    BitWidth        = "   << ib->bitWidth  << endl;
380                 os << "    VirtualCellSize = "   << ib->vc        << endl;
381                 os << "    DirectoryPath   = \"" << ib->dataDir   << "\"" << endl;
382                 os << "    Prefix          = \"" << ib->prefix    << "\"" << endl;
383                 os << "    Extension       = \"" << ib->extension << "\"" << endl;
384                 os << "    GatherMode      = \"" << (ib->isGather ? string("gathered") : string("
    distributed")) << "\"" << endl;
385                 os << "  }" << endl;
386                 os << "}" << endl;
387
388                 std::string filepath = m_targetDir + std::string("cellid.bcm");
389                 ofstream ofs( filepath.c_str() );
390                 if( !ofs ){
391                         Logger::Error("failed to open file (%s) .[%s:%d]\n", filepath.c_str(),
     __FILE__, __LINE__);
392                         return false;
393                 }
394
395                 ofs << os.str();
396                 return true;
397         }
```

5.2.3.7 bool BCMFileIO::BCMFileSaver::SaveIndexData (const std::string & *procName*, const std::string & *octName*) [private]

Data 情報ファイルを出力

引数

in	*procName*	プロセス情報ファイルの名前
in	*octName*	Octree ファイルの名前

戻り値

成功した場合 true, 失敗した場合 false

BCMFileSaver.cpp の 399 行目に定義があります。

参照先 BCMFileIO::Logger::Error(), BCMFileIO::IdxUnit::L0_scale, BCMFileIO::LB_CELLID, BCMFileIO::LB_SCALAR, BCMFileIO::IdxUnit::length, m_blockManager, m_comm, m_globalOrigin, m_globalRegion, m_idxBlockList, m_targetDir, m_unit, BCMFileIO::IdxUnit::V0_scale, BCMFileIO::IdxUnit::velocity.

参照元 SaveIndex().

```
400         {
401                 using namespace std;
402                 if( m_comm.Get_rank() != 0 ){ return true; }
403                 // search Data IdxBlocks
404                 vector<IdxBlock*> ibs;
405                 for(vector<IdxBlock>::iterator it = m_idxBlockList.begin(); it !=
    m_idxBlockList.end(); ++it){
406                         if( it->kind != LB_CELLID ){
407                                 ibs.push_back(&(*it));
408                         }
409                 }
410                 if( ibs.size() == 0) {
411                         return true;
412                 }
413
414                 Vec3i leafSize = m_blockManager.getSize();
415
416                 ostringstream os;
417                 os.setf(ios::scientific);
418                 os.precision(6);
419
420                 const char *typeStr[10] = {
421                         "Int8", "UInt8", "Int16", "UInt16", "Int32", "UInt32", "Int64", "UInt64", "Float32"
    , "Float64"
422                 };
423
424                 // write Data Information
425                 os << "Domain { " << endl;
426                 os << "  GlobalOrigin = " << m_globalOrigin << endl;
427                 os << "  GlobalRegion = " << m_globalRegion << endl;
428                 os << "}" << endl;
429                 os << endl;
430                 os << "BCMTree { " << endl;
431                 os << "  TreeFile  = \"" << octName  << "\"" << endl;
432                 os << "  ProcFile  = \"" << procName << "\"" << endl;
433                 os << "}" << endl;
434                 os << endl;
435                 os << "LeafBlock {" << endl;
436                 os << "  size = " << leafSize << endl << endl;
437                 os << "  Unit { " << endl;
438                 os << "    Length   = \"" << m_unit.length << "\"" << endl;
439                 os << "    L0       = "   << m_unit.L0_scale << endl;
440                 os << "    Velocity = \"" << m_unit.velocity << "\"" << endl;
441                 os << "    V0       = "  << m_unit.V0_scale << endl;
442                 os << "  }" << endl << endl;
443                 for(vector<IdxBlock*>::iterator it = ibs.begin(); it != ibs.end(); ++it){
444                         os << "  Data[@] { " << endl;
445                         os << "    name               = \"" << (*it)->name                          << "\" <
    < endl;
446                         os << "    NumberOfComponents = "   << ((*it)->kind ==
    LB_SCALAR ? 1 : 3) << endl;
447                         os << "    Type               = \"" << typeStr[(int)((*it)->dataType)]    << "\" <
    < endl;
448                         os << "    VirtualCellSize    = "   << (*it)->vc                           << endl;
449                         os << "    DirectoryPath      = \"" << (*it)->dataDir                       << "\" <
    < endl;
450                         os << "    Prefix             = \"" << (*it)->prefix                        << "\" <
```

```
    < endl;
451                         os << "    Extension          = \"" << (*it)->extension                    << "\"" <
    < endl;
452                         os << "    StepSubDirectory   = \"" << ((*it)->isStepSubDir ? string("true") :
    string("false")) << "\"" << endl;
453
454                         os << endl;
455                         unsigned int stepRange[3] = { (*it)->step.GetRangeMin(), (*it)->step.GetRangeMax(),
     (*it)->step.GetRangeInterval() };
456                         const vector<unsigned int>& stepAdds = (*it)->step.GetAddStepList();
457                         const vector<unsigned int>& stepSubs = (*it)->step.GetSubStepList();
458
459                         os << "    Step {" << endl;
460                         os << "      base = @range(" << stepRange[0] << "," << stepRange[1] << "," <<
    stepRange[2] << ")" << endl;
461                         if( stepAdds.size() != 0){
462                                 os << "      Add  = @list(" << stepAdds[0];
463                                 //for(vector<unsigned int>::const_iterator it = ++(stepAdds.begin()); it !=
     stepAdds.end(); ++it){ os << "," << (*it); }
464                                 for(size_t i = 1; i < stepAdds.size(); i++){ os << "," << stepAdds[i]; }
465                                 os << ")" << endl;
466                         }
467                         if( stepSubs.size() != 0){
468                                 os << "      Sub  = @list(" << stepSubs[0];
469                                 //for(vector<unsigned int>::const_iterator it = ++(stepSubs.begin()); it !=
     stepSubs.end(); ++it){ os << "," << (*it); }
470                                 for(size_t i = 1; i < stepSubs.size(); i++){ os << "," << stepSubs[i]; }
471                                 os << ")" << endl;
472                         }
473                         os << endl;
474                         os << "      Time   = " << (*it)->step.GetInitialTime() << endl;
475                         os << "      DeltaT = " << (*it)->step.GetDeltaT() << endl;
476                         os << "    }" <<endl;
477
478                         os << "  }" << endl << endl;
479                 }
480                 os << "}" << endl;
481
482                 std::string filepath = m_targetDir + std::string("data.bcm");
483                 ofstream ofs( filepath.c_str() );
484                 if( !ofs ){
485                         Logger::Error("failed to open file (%s) .[%s:%d]\n", filepath.c_str(),
    __FILE__, __LINE__);
486                         return false;
487                 }
488
489                 ofs << os.str();
490                 return true;
491         }
```

5.2.3.8 bool BCMFileIO::BCMFileSaver::SaveIndexProc (const std::string & *filepath,* const Partition & *part*) [private]

プロセス情報ファイルを出力

引数

in	*filepath*	プロセス情報ファイルの名前 (相対パス)
in	*part*	リーフブロックの 1 次元分割情報

戻り値

成功した場合 true, 失敗した場合 false

BCMFileSaver.cpp の 253 行目に定義があります。

参照先 BCMFileIO::Logger::Error(), m_comm, MPI_Comm_rank(), MPI_Comm_size(), MPI_COMM_WORLD.

参照元 SaveIndex().

```
254         {
255                 using namespace std;
256
257                 // Gather Hostnames
258                 char hostname[MPI_MAX_PROCESSOR_NAME + 1] = {0};
259                 int nameLen;
```

```
260                 MPI_Get_processor_name(hostname, &nameLen);
261                 hostname[nameLen] = '\0';
262                 nameLen++;
263 
264                 int* nameLenTable = NULL;
265                 if(m_comm.Get_rank() == 0 ){
266                         nameLenTable = new int[m_comm.Get_size()];
267                 }
268                 m_comm.Gather(&nameLen, 1, MPI::INT, nameLenTable, 1, MPI::INT, 0);
269 
270 
271                 char** hostnameList = NULL;
272                 if(m_comm.Get_rank() == 0){
273                         hostnameList = new char*[m_comm.Get_size()];
274                         hostnameList[0] = new char[nameLenTable[0]];
275                         memset(hostnameList[0], 0, sizeof(char) * nameLenTable[0]);
276                         memcpy(hostnameList[0], hostname, sizeof(char) * nameLen);
277 
278                         for(int i = 1; i < m_comm.Get_size(); i++){
279                                 hostnameList[i] = new char[nameLenTable[i]];
280                                 memset(hostnameList[i], 0, sizeof(char) * nameLenTable[i]);
281                                 m_comm.Recv(hostnameList[i], nameLenTable[i], MPI::CHAR, i, i);
282                         }
283                         delete [] nameLenTable;
284                 }else{
285                         m_comm.Send(hostname, nameLen, MPI::CHAR, 0,
    m_comm.Get_rank());
286                 }
287 
288                 if( m_comm.Get_rank() != 0){
289                         return true;
290                 }
291 
292                 stringstream os;
293                 os.setf(ios::scientific);
294                 os.precision(6);
295 
296                 int  NumberOfRank  = 0;
297                 int  NumberOfGroup = 1;
298                 int  RankID        = 0;
299                 int  GroupID       = 0;
300 
301                 MPI_Comm_size(MPI_COMM_WORLD, &NumberOfRank);
302                 MPI_Comm_rank(MPI_COMM_WORLD, &RankID);
303 
304                 os << "MPI {" << endl;
305                 os << "  NumberOfRank   = " << NumberOfRank   << endl;
306                 os << "  NumberOfGroup  = " << NumberOfGroup  << endl;
307                 os << "  RankID         = " << RankID         << endl;
308                 os << "  GroupID        = " << GroupID        << endl;
309                 os << "}" << endl;
310                 os << endl;
311                 os << "Process { " << endl;
312                 for(int proc = 0; proc < m_comm.Get_size(); proc++){
313                         os << "  Rank[@] { " << endl;
314                         os << "    ID          = "   << proc << endl;
315                         os << "    HostName    = \"" << hostnameList[proc] << "\"" << endl;
316                         os << "    BlockRange = @range(" << part.getStart(proc) <<  "," << part.getEnd(proc
    ) - 1 << ")" << endl;
317                         os << "  }" << endl << endl;
318                 }
319                 os << "}" << endl;
320 
321                 ofstream ofs(filepath.c_str());
322                 if( !ofs ){
323                         Logger::Error("failed to open file (%s) .[%s:%d]\n", filepath.c_str(),
     __FILE__, __LINE__);
324                         for(int i = 0; i < m_comm.Get_size(); i++) delete [] hostnameList[i];
325                         delete hostnameList;
326                         return false;
327                 }
328 
329                 ofs << os.str();
330 
331                 for(int i = 0; i < m_comm.Get_size(); i++) delete [] hostnameList[i];
332                 delete hostnameList;
333 
334                 return true;
335         }
```

5.2.3.9 bool BCMFileIO::BCMFileSaver::SaveLeafBlock (const char * *name,* unsigned int *step* = 0)

リーフブロックファイルを出力

引数

in	*name*	系の名称 (Register した際に設定した名前)
in	*step*	タイムステップ

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

ファイル出力する対象がCellID の場合、step は無視される.

BCMFileSaver.cpp の 192 行目に定義があります。

参照先 BCMFileIO::FileSystemUtil::CreateDirectory(), BCMFileIO::Logger::Error(), BCMFileIO::IdxBlock::find(), GetCellIDBlock(), BCMFileIO::LB_CELLID, m_blockManager, m_comm, m_idxBlockList, BCMFileIO::ErrorUtil::reduceError(), BCMFileIO::LeafBlockSaver::SaveCellID(), BCMFileIO::LeafBlockSaver::SaveData().

```
193         {
194                 using namespace std;
195
196                 bool err = false;
197
198                 IdxBlock *ib = IdxBlock::find(m_idxBlockList, name);
199
200                 if( ib == NULL ){
201                         Logger::Error("%s is not registerd. [%s:%d]\n", name, __FILE__,
    __LINE__);
202                         err = true;
203                 }
204
205                 if( ErrorUtil::reduceError(err) ){ return false; }
206                 err = false;
207
208                 if( ib->kind == LB_CELLID )
209                 {
210                         string lbdir = ib->rootDir + ib->dataDir;
211                         if( ib->isGather ){
212                                 if( m_comm.Get_rank() == 0 ) err = !
    FileSystemUtil::CreateDirectory(lbdir, lbdir.find("/") == 0 ? true : false);
213                         }else{
214                                 err = !FileSystemUtil::CreateDirectory(lbdir
    , lbdir.find("/") == 0 ? true : false);
215                         }
216
217                         if( ErrorUtil::reduceError(err) ){
218                                 Logger::Error("Cannot Create Output Directory (%s) [%s:%d]\n",
     lbdir.c_str(), __FILE__, __LINE__);
219                                 return false;
220                         }
221
222                         unsigned char *data = GetCellIDBlock(ib,
    m_blockManager);
223                         if( data == NULL ){
224                                 err = true;
225                         } else {
226 #ifdef ENABLE_RLE_ENCODE
227                                 err = !LeafBlockSaver::SaveCellID(
    m_comm, ib, m_blockManager.getSize(), m_blockManager.getNumBlock(), data,
     true);
228 #else
229                                 err = !LeafBlockSaver::SaveCellID(
    m_comm, ib, m_blockManager.getSize(), m_blockManager.getNumBlock(), data,
     false);
230 #endif // ENABLE_RLE_ENCODE
231                                 delete [] data;
232                         }
233
234                         if( ErrorUtil::reduceError(err) ){
235                                 Logger::Error("Save Leaf Block (CellID) [%s:%d]\n", __FILE__,
    __LINE__);
236                                 return false;
237                         }
238                 }
239                 else
240                 {
241                         err = !LeafBlockSaver::SaveData(
```

```
    m_comm, ib, m_blockManager, step);
242
243                         if( ErrorUtil::reduceError(err) ){
244                                 Logger::Error("Save Leaf Block (Scalar) [%s:%d]\n", __FILE__,
    __LINE__);
245                                 return false;
246                         }
247                 }
248
249                 return true;
250
251         }
```

5.2.3.10 bool BCMFileIO::BCMFileSaver::SaveOctree (const std::string & *filepath*, const BCMOctree * *octree*) [private]

Octree ファイルを出力

引数

in	*filepath*	Octree ファイル名 (相対パス)
in	*octree*	Octree

戻り値

成功した場合 true, 失敗した場合 false

BCMFileSaver.cpp の 518 行目に定義があります。

参照先 BCMFileIO::Logger::Error(), BCMFileIO::OctHeader::identifier, m_comm, m_globalOrigin, m_globalRegion, OCTREE_FILE_IDENTIFIER, Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z.

参照元 Save().

```
519         {
520                 using namespace std;
521
522                 if( m_comm.Get_rank() != 0 ){
523                         return true;
524                 }
525
526                 FILE *fp = NULL;
527                 if( (fp = fopen(filepath.c_str(), "wb")) == NULL ){
528                         Logger::Error("faild open file (%s) .[%s:%d]\n", filepath.c_str(),
    __FILE__, __LINE__);
529                         return false;
530                 }
531
532                 OctHeader header;
533                 header.identifier = OCTREE_FILE_IDENTIFIER;
534
535                 const RootGrid* rootGrid = octree->getRootGrid();
536
537                 header.org[0]      = m_globalOrigin.x;
538                 header.org[1]      = m_globalOrigin.y;
539                 header.org[2]      = m_globalOrigin.z;
540                 header.rgn[0]      = m_globalRegion.x;
541                 header.rgn[1]      = m_globalRegion.y;
542                 header.rgn[2]      = m_globalRegion.z;
543                 header.rootDims[0] = rootGrid->getSizeX();
544                 header.rootDims[1] = rootGrid->getSizeY();
545                 header.rootDims[2] = rootGrid->getSizeZ();
546
547                 header.numLeaf = octree->getNumLeafNode();
548
549                 const vector<Node*> nodes  = octree->getLeafNodeArray();
550                 vector<Pedigree>    pedigs;
551                 pedigs.reserve(octree->getNumLeafNode());
552
553                 int maxLevel = 0;
554                 for(vector<Node*>::const_iterator it = nodes.begin(); it != nodes.end(); ++it){
555                         const Node* n = *it;
556                         maxLevel = n->getLevel() > maxLevel ? n->getLevel() : maxLevel;
557                         pedigs.push_back(n->getPedigree());
```

```
558                 }
559
560                 header.maxLevel = maxLevel;
561
562                 fwrite(&header, sizeof(header), 1, fp);
563                 fwrite(&pedigs[0], sizeof(Pedigree), pedigs.size(), fp);
564
565                 fclose(fp);
566
567                 return true;
568         }
```

5.2.3.11 bool BCMFileIO::BCMFileSaver::SetUnit (const IdxUnit & *unit*)

出力対象データの単位系設定

引数

in	*unit*	単位系

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

設定を省略した場合、デフォルト値で動作

BCMFileSaver.cpp の 150 行目に定義があります。

参照先 m_unit.

```
151         {
152                 m_unit = unit;
153                 return true;
154         }
```

5.2.4 メンバ詳解

5.2.4.1 BlockManager& BCMFileIO::BCMFileSaver::m_blockManager [private]

ブロックマネージャ

BCMFileSaver.h の 179 行目に定義があります。

参照元 SaveIndexCellID(), SaveIndexData(), SaveLeafBlock().

5.2.4.2 const MPI::Intracomm& BCMFileIO::BCMFileSaver::m_comm [private]

MPI コミュニケータ

BCMFileSaver.h の 180 行目に定義があります。

参照元 Save(), SaveIndex(), SaveIndexCellID(), SaveIndexData(), SaveIndexProc(), SaveLeafBlock(), SaveOctree().

5.2.4.3 const Vec3d BCMFileIO::BCMFileSaver::m_globalOrigin [private]

計算空間の起点座標

BCMFileSaver.h の 182 行目に定義があります。

参照元 SaveIndexCellID(), SaveIndexData(), SaveOctree().

5.2.4.4 const Vec3d BCMFileIO::BCMFileSaver::m_globalRegion [private]

計算空間全体の領域サイズ

BCMFileSaver.h の 183 行目に定義があります。

参照元 SaveIndexCellID(), SaveIndexData(), SaveOctree().

5.2.4.5 std::vector<IdxBlock> BCMFileIO::BCMFileSaver::m_idxBlockList [private]

登録されたブロック情報リスト

BCMFileSaver.h の 186 行目に定義があります。

参照元 RegisterCellIDInformation(), RegisterDataInformation(), SaveIndexCellID(), SaveIndexData(), SaveLeafBlock().

5.2.4.6 const BCMOctree* BCMFileIO::BCMFileSaver::m_octree [private]

BCMOctree.

BCMFileSaver.h の 181 行目に定義があります。

参照元 Save().

5.2.4.7 std::string BCMFileIO::BCMFileSaver::m_targetDir [private]

ファイル出力ターゲットディレクトリ名

BCMFileSaver.h の 185 行目に定義があります。

参照元 BCMFileSaver(), RegisterCellIDInformation(), RegisterDataInformation(), Save(), SaveIndex(), SaveIndexCellID(), SaveIndexData().

5.2.4.8 IdxUnit BCMFileIO::BCMFileSaver::m_unit [private]

単位系

BCMFileSaver.h の 184 行目に定義があります。

参照元 BCMFileSaver(), SaveIndexCellID(), SaveIndexData(), SetUnit().

このクラス詳解は次のファイルから抽出されました:

- BCMFileSaver.h
- BCMFileSaver.cpp

5.3 BCMFileIO::BCMRLE クラス

ランレングスによる圧縮/展開ライブラリ

`#include <BCMRLE.h>`

静的公開メンバ関数

- template<typename rluint_t , typename runlen_t >
  static unsigned char * Encode (const rluint_t *source, const size_t sourceSize, size_t *destSize)
- template<typename rluint_t , typename runlen_t >
  static rluint_t * Decode (const unsigned char *source, const size_t sourceSize, const size_t destSize)

5.3.1 詳解

ランレングスによる圧縮/展開ライブラリ

BCMRLE.h の 20 行目に定義があります。

5.3.2 関数詳解

5.3.2.1 template<typename rluint_t , typename runlen_t > static rluint_t* BCMFileIO::BCMRLE::Decode (const unsigned char * *source*, const size_t *sourceSize*, const size_t *destSize*) [inline], [static]

RLE 展開

引数

in	*source*	入力データの先頭ポインタ (RLE 圧縮符号)
in	*sourceSize*	入力データのサイズ (Byte 単位で指定)
out	*destSize*	出力データのサイズ (Byte 単位で指定)

戻り値

RLE 圧縮符号を展開したデータのの先頭ポインタ

覚え書き

return されたポインタは適宜解放 (delete) してください.

BCMRLE.h の 95 行目に定義があります。

参照先 BCMFileIO::ALIGNMENT.

```
96                 {
97                         #ifdef __GNUC__
98                         #pragma pack(push, 1)
99                         #define ALIGNMENT __attribute__((packed))
100                         #else
101                         #pragma pack(1)
102                         #define ALIGNMENT
103                         #endif // __GNUC__
104                                 // RLE 符号を簡単に走査するためアライメントを無効にした構造体を定義
105                                 struct DR{
```

```
106                                     rluint_t d;
107                                     runlen_t len;
108                             } ALIGNMENT;
109                     #ifdef __GNUC__
110                     #pragma pack(pop)
111                     #else  // __GNUC__
112                     #pragma pack()
113                     #endif // __GNUC__
114
115                     size_t endData = destSize / sizeof(rluint_t);
116
117                     rluint_t* dest = new rluint_t[endData];
118
119                     const DR* pdr = reinterpret_cast<const DR*>(source);
120                     size_t num = sourceSize / sizeof(DR);
121
122                     size_t cnt = 0;
123                     for(size_t i = 0; i < num; i++){
124                             for(runlen_t l = 0; l < pdr->len; l++){
125                                     if(cnt >= endData){
126                                             delete [] dest;
127                                             return NULL;
128                                     }
129                                     dest[cnt] = pdr->d;
130                                     cnt++;
131                             }
132                             pdr++;
133                     }
134
135                     return dest;
136             }
```

5.3.2.2 template<typename rluint_t , typename runlen_t > static unsigned char* BCMFileIO::BCMRLE::Encode (const rluint_t * *source*, const size_t *sourceSize*, size_t * *destSize*) [inline], [static]

RLE 圧縮

引数

in	*source*	入力データの先頭ポインタ
in	*sourceSize*	入力データのサイズ (Byte 単位で指定)
out	*destSize*	出力データのサイズ (Byte 単位で指定)

戻り値

RLE 圧縮符号の先頭ポインタ．エラーの場合NULL を返す

覚え書き

return されたポインタは適宜解放 (delete) してください．

BCMRLE.h の 32 行目に定義があります。

参照先 BCMFileIO::ALIGNMENT.

```
33              {
34                      #ifdef __GNUC__
35                      #pragma pack(push, 1)
36                      #define ALIGNMENT __attribute__((packed))
37                      #else
38                      #pragma pack(1)
39                      #define ALIGNMENT
40                      #endif // __GNUC__
41                              // RLE 符号を簡単に走査するためアライメントを無効にした構造体を定義
42                              struct DR{
43                                      rluint_t d;
44                                      runlen_t len;
45                              } ALIGNMENT;
46                      #ifdef __GNUC__
47                      #pragma pack(pop)
48                      #else  // __GNUC__
```

```
49                      #pragma pack()
50                      #endif // __GNUC__
51
52                      const runlen_t maxCount = (runlen_t)~0;
53                      const size_t endData = sourceSize / sizeof(rluint_t);
54
55                      const rluint_t* pSrc = source;
56
57                      size_t maxSize = sourceSize * sizeof(rluint_t) + sourceSize * sizeof(runlen_t);
58                      unsigned char*  dest = new unsigned char[maxSize];
59                      memset(dest, 0, maxSize * sizeof(unsigned char));
60
61                      DR* pdr = reinterpret_cast<DR*>(dest);
62
63                      pdr[0].d   = pSrc[0];
64                      pdr[0].len = 1;
65                      size_t cnt = 0;
66                      for(size_t i = 1; i < endData; i++){
67                              const rluint_t d = pSrc[i];
68                              if( pdr[cnt].d != d || pdr[cnt].len == maxCount )
69                              {
70                                      cnt++;
71                                      pdr[cnt].d = d;
72                                      pdr[cnt].len = 1;
73                              }
74                              else
75                              {
76                                      pdr[cnt].len++;
77                              }
78                      }
79
80                      *destSize = (cnt + 1) * sizeof(DR);
81
82                      return dest;
83              }
```

このクラス詳解は次のファイルから抽出されました:

- BCMRLE.h

5.4 BCMFileIO::BitVoxel クラス

ビットボクセル圧縮/展開ライブラリ

`#include <BitVoxel.h>`

公開型

- typedef unsigned int bitVoxelCell
  ビットボクセル型の定義

公開メンバ関数

- BitVoxel ()
  コンストラクタ
- ~BitVoxel ()
  デストラクタ

静的公開メンバ関数

- static size_t GetSize (const size_t sourceSize, const unsigned char bitWidth)
- static bitVoxelCell * Compress (size_t *bitVoxelSize, const size_t voxelSize, const unsigned char *voxel, const unsigned char bitWidth)

- static unsigned char * Decompress (const size_t voxelSize, const bitVoxelCell *bitVoxel, const unsigned char bitWidth)

5.4.1 詳解

ビットボクセル圧縮/展開ライブラリ

BitVoxel.h の 23 行目に定義があります。

5.4.2 型定義メンバ詳解

5.4.2.1 typedef unsigned int BCMFileIO::BitVoxel::bitVoxelCell

ビットボクセル型の定義

BitVoxel.h の 27 行目に定義があります。

5.4.3 構築子と解体子

5.4.3.1 BCMFileIO::BitVoxel::BitVoxel ( )

コンストラクタ

BitVoxel.cpp の 22 行目に定義があります。

```
23          {
24          }
```

5.4.3.2 BCMFileIO::BitVoxel::~BitVoxel ( )

デストラクタ

BitVoxel.cpp の 26 行目に定義があります。

```
27          {
28          }
```

5.4.4 関数詳解

5.4.4.1 bitVoxelCell * BCMFileIO::BitVoxel::Compress (size_t * *bitVoxelSize,* const size_t *voxelSize,* const unsigned char * *voxel,* const unsigned char *bitWidth*) [static]

ビットボクセル圧縮

引数

out	*bitVoxelSize*	出力ビットボクセルサイズ
in	*boxelSize*	入力ボクセルサイズ
in	*voxel*	入力ボクセルの先頭ポインタ
in	*bitWidth*	ビット幅

戻り値

ビットボクセルの先頭ポインタ

覚え書き

return されたポインタは適宜解放 (delete) してください.

BitVoxel.cpp の 36 行目に定義があります。

```
37         {
38                 const unsigned char vox_per_cell = (sizeof(bitVoxelCell) * 8) / bitWidth;
39                 size_t bsz  = voxelSize / vox_per_cell + (voxelSize % vox_per_cell == 0 ? 0 : 1);
40
41                 bitVoxelCell* bitVoxel = new bitVoxelCell[bsz];
42                 memset(bitVoxel, 0, sizeof(bitVoxelCell) * bsz);
43
44                 unsigned char mask = 0;
45                 for(int i = 0; i < bitWidth; i++) mask += (1 << i);
46
47                 for(size_t i = 0; i < voxelSize; i++){
48                         size_t       cellIdx =  i / vox_per_cell;
49                         unsigned int bitIdx  = (i % vox_per_cell) * bitWidth;
50
51                         unsigned char c = voxel[i];
52
53                         bitVoxel[cellIdx] += (c & mask) << bitIdx;
54                 }
55
56                 *bitVoxelSize = bsz;
57                 return bitVoxel;
58
59         }
```

5.4.4.2 unsigned char * BCMFileIO::BitVoxel::Decompress (const size_t *voxelSize*, const bitVoxelCell * *bitVoxel*, const unsigned char *bitWidth*) [static]

ビットボクセル展開

引数

in	*bitVoxelSize*	ボクセルサイズ (展開後のボクセル数)
in	*bitVoxel*	入力ビットボクセル
in	*bitWidth*	ビット幅

戻り値

展開されたボクセルの先頭ポインタ

覚え書き

return されたポインタは適宜解放 (delete) してください.

BitVoxel.cpp の 61 行目に定義があります。

```
62          {
63                  const unsigned char vox_per_cell = (sizeof(bitVoxelCell) * 8) / bitWidth;
64
65                  unsigned char* voxel = new unsigned char[voxelSize];
66                  memset(voxel, 0, sizeof(unsigned char) * voxelSize);
67
68                  unsigned char mask = 0;
69                  for(int i = 0; i < bitWidth; i++) mask += (1 << i);
70
71                  for(size_t i = 0; i < voxelSize; i++){
72                          size_t       cellIdx =  i / vox_per_cell;
73                          unsigned int bitIdx  = (i % vox_per_cell) * bitWidth;
74
75                          voxel[i] = (bitVoxel[cellIdx] >> bitIdx) & mask;
76                  }
77
78                  return voxel;
79          }
```

5.4.4.3 size_t BCMFileIO::BitVoxel::GetSize (const size_t *sourceSize*, const unsigned char *bitWidth*) [static]

ボクセルをビットボクセル化した場合のビットボクセルサイズを出力

引数

in	*sourceSize*	ボクセル数
in	*bitWidth*	ビット幅

戻り値

ビットボクセルサイズ

覚え書き

ビットボクセルサイズはバイト単位ではない.

BitVoxel.cpp の 30 行目に定義があります。

```
31          {
32                  const unsigned char vox_per_cell = (sizeof(bitVoxelCell) * 8) / bitWidth;
33                  return (voxelSize / vox_per_cell + (voxelSize % vox_per_cell == 0 ? 0 : 1 ));
34          }
```

このクラス詳解は次のファイルから抽出されました:

- BitVoxel.h
- BitVoxel.cpp

5.5 BCMFileIO::LeafBlockLoader::CellIDCapsule 構造体

グリッドヘッダとデータを一括りにした構造体

`#include <LeafBlockLoader.h>`

BCMFileIO::LeafBlockLoader::CellIDCapsule 連携図

公開メンバ関数

- CellIDCapsule ()

公開変数類

- LBCellIDHeader header
  リーフブロックのグリッドヘッダ
- unsigned char * data
  リーフブロックデータ

5.5.1 詳解

グリッドヘッダとデータを一括りにした構造体

LeafBlockLoader.h の 35 行目に定義があります。

5.5.2 構築子と解体子

5.5.2.1 BCMFileIO::LeafBlockLoader::CellIDCapsule::CellIDCapsule ( ) [inline]

LeafBlockLoader.h の 39 行目に定義があります。

```
39 : data(NULL){}
```

5.5.3 メンバ詳解

5.5.3.1 unsigned char* BCMFileIO::LeafBlockLoader::CellIDCapsule::data

リーフブロックデータ

LeafBlockLoader.h の 38 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::DecompCellIDData(), BCMFileIO::LeafBlockLoader::LoadCellID(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather().

5.5.3.2 LBCellIDHeader BCMFileIO::LeafBlockLoader::CellIDCapsule::header

リーフブロックのグリッドヘッダ

LeafBlockLoader.h の 37 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::DecompCellIDData(), BCMFileIO::LeafBlockLoader::LoadCellID(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather().

この構造体詳解は次のファイルから抽出されました:

- LeafBlockLoader.h

5.6 BCMFileIO::DirUtil クラス

ディレクトリ操作ユーティリティ

`#include <DirUtil.h>`

公開メンバ関数

- DirUtil (const char *dirname)
  コンストラクタ
- unsigned int GetFileCount () const
  ディレクトリ内のファイル数を取得
- unsigned int GetDirCount () const
  ディレクトリ内のディレクトリ数を取得
- const std::string & GetFile (unsigned int i) const
  ディレクトリ内の指定インデックスのファイル名を取得
- const std::string & GetDir (unsigned int i) const
  ディレクトリ内の指定インデックスのディレクトリ名を取得
- std::string GetFilePath (const char *filename) const
  入力ファイル名をフルパスにして返却
- bool IsOpened () const
  ディレクトリがオープン中か返却
- const std::string & GetPath () const
  パス名を返却
- const std::string & GetName () const
  ディレクトリ名を返却
- bool CreateDir (const char *dirname)
  ディレクトリ作成

非公開メンバ関数

- std::string parseFilename (const char *fpath, const char *splitChar)
  指定ファイルパスから最後の区切り文字位置で分割した文字列を返却

非公開変数類

- std::string m_dirname
- std::string m_dirpath
- bool m_opened
- std::vector< std::string > m_files
- std::vector< std::string > m_dirs

5.6.1 詳解

ディレクトリ操作ユーティリティ

DirUtil.h の 23 行目に定義があります。

5.6.2 構築子と解体子

5.6.2.1 BCMFileIO::DirUtil::DirUtil (const char * *dirname*)

コンストラクタ

コンストラクタ

引数

in	ディレクトリ名	

DirUtil.cpp の 26 行目に定義があります。

参照先 m_dirname, m_dirpath, m_dirs, m_files, m_opened, parseFilename().

```
27         {
28                 m_opened = false;
29                 m_dirpath = std::string(dirname);
30
31                 m_dirname = parseFilename(dirname,  "/");
32                 DIR* pDir = opendir(dirname);
33                 if (!pDir)
34                         return;
35
36                 dirent* pEnt = readdir(pDir);
37             while (pEnt)
38                 {
39                 if ( strcmp( pEnt->d_name, "." ) &&
40                      strcmp( pEnt->d_name, ".." ) )
41                         {
42                                 std::string wPathName = std::string(dirname) + std::string("/") +
    std::string(pEnt->d_name);
43                                 struct stat wStat;
44                     if ( stat( wPathName.c_str(), &wStat ) )
45                         break;
46
47                     if ( S_ISDIR( wStat.st_mode ) )
48                                         m_dirs.push_back(pEnt->d_name);
49                     else
50                         m_files.push_back(pEnt->d_name);
51                 }
52                         pEnt = readdir(pDir);
53             }
54                 closedir(pDir);
55                 m_opened = true;
56         }
```

5.6.3 関数詳解

5.6.3.1 bool BCMFileIO::DirUtil::CreateDir (const char * *dirname*)

ディレクトリ作成

ディレクトリ作成

引数

in	ディレクトリ名	

戻り値

true: 作成成功、もしくは既に同一ディレクトリあり、false: 作成失敗、もしくは既に同一ファイル名あり

DirUtil.cpp の 115 行目に定義があります。

参照先 GetFilePath(), IsOpened(), m_dirs, m_files.

```
116         {
117                 std::string dirpath = GetFilePath(dirname);
```

```
118                 std::vector<std::string>::iterator it = std::find(m_files.begin(),
    m_files.end(), dirname);
119                 if (it != m_files.end())
120                         return false;
121                 it = std::find(m_dirs.begin(), m_dirs.end(), dirname);
122                 if (it != m_dirs.end())
123                         return true;
124
125                 const mode_t mode = S_IRWXU | S_IRGRP | S_IXGRP | S_IROTH | S_IXOTH;
126                 if ( mkdir(dirpath.c_str(), mode) != 0 )
127                 {
128                         usleep(10000);
129                         DirUtil check(dirpath.c_str());
130                         return check.IsOpened();
131                 }
132
133                 return true;
134         }
```

5.6.3.2 const std::string & BCMFileIO::DirUtil::GetDir (unsigned int *i*) const

ディレクトリ内の指定インデックスのディレクトリ名を取得

ディレクトリ内の指定インデックスのディレクトリ名を取得

引数

in	**インデックス**	

戻り値

ディレクトリ名. インデックスがオーバーした場合は nullstring を返却

DirUtil.cpp の 87 行目に定義があります。

参照先 m_dirs.

```
88         {
89                 static std::string nullstring;
90                 if (i >= m_dirs.size())
91                         return nullstring;
92
93                 return m_dirs[i];
94         }
```

5.6.3.3 unsigned int BCMFileIO::DirUtil::GetDirCount ( ) const

ディレクトリ内のディレクトリ数を取得

ディレクトリ内のディレクトリ数を取得

戻り値

ディレクトリ数

DirUtil.cpp の 71 行目に定義があります。

参照先 m_dirs.

```
72         {
73                 return static_cast<unsigned int>(m_dirs.size());
74         }
```

5.6.3.4 const std::string & BCMFileIO::DirUtil::GetFile (unsigned int *i*) const

ディレクトリ内の指定インデックスのファイル名を取得

ディレクトリ内の指定インデックスのファイル名を取得

引数

in	**インデックス**	

戻り値

ファイル名. インデックスがオーバーした場合は nullstring を返却

DirUtil.cpp の 77 行目に定義があります。

参照先 m_files.

```
78         {
79                 static std::string nullstring;
80                 if (i >= m_files.size())
81                         return nullstring;
82
83                 return m_files[i];
84         }
```

5.6.3.5 unsigned int BCMFileIO::DirUtil::GetFileCount ( ) const

ディレクトリ内のファイル数を取得

ディレクトリ内のファイル数を取得

戻り値

ファイル数

DirUtil.cpp の 65 行目に定義があります。

参照先 m_files.

```
66         {
67                 return static_cast<unsigned int>(m_files.size());
68         }
```

5.6.3.6 std::string BCMFileIO::DirUtil::GetFilePath (const char * *filename*) const

入力ファイル名をフルパスにして返却

入力ファイル名をフルパスにして返却

引数

in	**ファイル名**	

戻り値

フルパスのファイル名

DirUtil.cpp の 97 行目に定義があります。

参照先 m_dirpath.

参照元 CreateDir().

```
98          {
99                  return m_dirpath + "/" + filename;
100         }
```

5.6.3.7 const std::string & BCMFileIO::DirUtil::GetName ( ) const

ディレクトリ名を返却

ディレクトリ名を返却

戻り値

ディレクトリ名

DirUtil.cpp の 109 行目に定義があります。

参照先 m_dirname.

```
110         {
111                 return m_dirname;
112         }
```

5.6.3.8 const std::string & BCMFileIO::DirUtil::GetPath ( ) const

パス名を返却

パス名を返却

戻り値

パス名

DirUtil.cpp の 103 行目に定義があります。

参照先 m_dirpath.

```
104         {
105                 return m_dirpath;
106         }
```

5.6.3.9 bool BCMFileIO::DirUtil::IsOpened ( ) const

ディレクトリがオープン中か返却

ディレクトリがオープン中か返却

戻り値

true: オープン中、false: クローズ中

DirUtil.cpp の 59 行目に定義があります。

参照先 m_opened.

参照元 CreateDir().

```
60          {
61                  return m_opened;
62          }
```

5.6.3.10 std::string BCMFileIO::DirUtil::parseFilename (const char * *fpath*, const char * *splitChar*) [private]

指定ファイルパスから最後の区切り文字位置で分割した文字列を返却

指定ファイルパスから最後の区切り文字位置で分割した文字列を返却

引数

in	**ファイルパス**	
in	**区切り文字**	

戻り値

分割後の文字列

DirUtil.cpp の 137 行目に定義があります。

参照元 DirUtil().

```
138         {
139                 std::string f(fpath);
140                 size_t p = f.rfind(splitChar);
141                 if (p != std::string::npos)
142                         f.erase(f.begin(), f.begin() + p + 1);
143                 return f;
144         }
```

5.6.4 メンバ詳解

5.6.4.1 std::string BCMFileIO::DirUtil::m_dirname [private]

DirUtil.h の 105 行目に定義があります。

参照元 DirUtil(), GetName().

5.6.4.2 std::string BCMFileIO::DirUtil::m_dirpath [private]

DirUtil.h の 106 行目に定義があります。

参照元 DirUtil(), GetFilePath(), GetPath().

5.6.4.3 std::vector<std::string> BCMFileIO::DirUtil::m_dirs [private]

DirUtil.h の 109 行目に定義があります。

参照元 CreateDir(), DirUtil(), GetDir(), GetDirCount().

5.6.4.4 std::vector<std::string> BCMFileIO::DirUtil::m_files [private]

DirUtil.h の 108 行目に定義があります。

参照元 CreateDir(), DirUtil(), GetFile(), GetFileCount().

5.6.4.5 bool BCMFileIO::DirUtil::m_opened [private]

DirUtil.h の 107 行目に定義があります。

参照元 DirUtil(), IsOpened().

このクラス詳解は次のファイルから抽出されました:

- DirUtil.h
- DirUtil.cpp

5.7 BCMFileIO::ErrorUtil クラス

エラー処理関連のユーティリティ

`#include <ErrorUtil.h>`

静的公開メンバ関数

- static bool reduceError (const bool err, MPI::Intracomm &comm=MPI::COMM_WORLD)

5.7.1 詳解

エラー処理関連のユーティリティ

ErrorUtil.h の 22 行目に定義があります。

5.7.2 関数詳解

5.7.2.1 bool BCMFileIO::ErrorUtil::reduceError (const bool *err*, MPI::Intracomm & *comm* = MPI::COMM_WORLD) [static]

全プロセスに対しエラー情報を配信

引数

in	*err*	エラーがある場合 true を入力
in	*comm*	MPI コミュニケータ

戻り値

1 プロセスでもエラーがある場合 true を返す．全プロセスでエラーが無い場合 false

ErrorUtil.cpp の 22 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::BCMFileLoader(), BCMFileIO::BCMFileLoader::CreateLeafBlock(), BCMFileIO::BCMFileLoader::LoadAdditionalIndex(), BCMFileIO::BCMFileLoader::LoadLeafBlock(), BCMFileIO::BCMFileLoader::LoadOctree(), BCMFileIO::BCMFileSaver::Save(), BCMFileIO::BCMFileSaver::SaveIndex(), BCMFileIO::BCMFileSaver::SaveLeafBlock().

```
22                                                                      {
23                 int ierr = err ? 1 : 0;
24                 comm.Allreduce(&ierr, &ierr, 1, MPI::INT, MPI::BOR);
25                 return ierr == 0 ? false : true;
26         }
```

このクラス詳解は次のファイルから抽出されました:

- ErrorUtil.h
- ErrorUtil.cpp

5.8 BCMFileIO::PartitionMapper::FDIDList 構造体

ファイルID とファイル内のデータID リスト構造体

`#include <PartitionMapper.h>`

公開変数類

- int FID
  FID.
- std::vector< int > FDIDs
  FDID **リスト**

5.8.1 詳解

ファイルID とファイル内のデータID リスト構造体

PartitionMapper.h の 31 行目に定義があります。

5.8.2 メンバ詳解

5.8.2.1 std::vector<int> BCMFileIO::PartitionMapper::FDIDList::FDIDs

FDID リスト

PartitionMapper.h の 34 行目に定義があります。

5.8.2.2 int BCMFileIO::PartitionMapper::FDIDList::FID

FID.

PartitionMapper.h の 33 行目に定義があります。

この構造体詳解は次のファイルから抽出されました:

- PartitionMapper.h

5.9 BCMFileIO::FileSystemUtil クラス

ファイル操作関連ユーティリティ

`#include <FileSystemUtil.h>`

静的公開メンバ関数

- static std::string ConvertPath (const std::string &path)
- static std::string GetDirectory (const std::string &path)
- static std::string GetFilePrefix (const std::string &path)
- static std::string FixDirectoryPath (const std::string &dir)
- static bool CreateDirectory (const std::string &path, bool absolutePath=false)
- static bool split (const std::string &input, const char delimiter, std::vector< std::string > &output)

5.9.1 詳解

ファイル操作関連ユーティリティ

FileSystemUtil.h の 32 行目に定義があります。

5.9.2 関数詳解

5.9.2.1 static std::string BCMFileIO::FileSystemUtil::ConvertPath (const std::string & *path*) [inline], [static]

ファイルパスの "\\" を "/" に変換

引数

in	*path*	ファイルパス

戻り値

"/"に変換されたファイルパス

FileSystemUtil.h の 40 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::BCMFileLoader(), GetDirectory(), GetFilePrefix(), BCMFileIO::BCMFileLoader::LoadAdditionalIndex(), BCMFileIO::BCMFileLoader::LoadIndex().

```
41                 {
42                         std::string r = path;
43                         size_t p = r.find("\\");
44                         while (p != std::string::npos)
45                         {
46                                 *(r.begin() + p) = '/';
47                                 p = r.find("\\");
48                         }
49                         return r;
50                 }
```

5.9.2.2 static bool BCMFileIO::FileSystemUtil::CreateDirectory (const std::string & *path*, bool *absolutePath* = false) [inline], [static]

path で指定したディレクトリを作成 (mkdir -p 相当)

引数

in	*path*	ディレクトリ名
in	*absolutePath*	true: ルートディレクトリ (/) から作成、false: 相対パスとして作成

戻り値

true: 作成成功、false: 作成失敗

FileSystemUtil.h の 107 行目に定義があります。

参照先 BCMFileIO::Logger::Error(), split().

参照元 BCMFileIO::BCMFileSaver::Save(), BCMFileIO::BCMFileSaver::SaveLeafBlock().

```
108                 {
109
110                         std::vector<std::string> dirList;
111                         split(path, '/', dirList);
112
113                         std::string npath;
114                         if( absolutePath ){
115                                 npath = std::string("/");
116                         }else{
117                                 npath = std::string("./");
118                         }
119
120                         for(std::vector<std::string>::iterator it = dirList.begin(); it != dirList.end(); +
    +it){
121                                 DirUtil dir(npath.c_str());
122
123                                 if( *it == std::string(".")  ){ continue; }
124                                 if( *it == std::string("..") ){ npath += *it + std::string("/"); continue;
    }
125
126                                 if( !dir.CreateDir(it->c_str()) ){
127                                         Logger::Error("cannnot create directory : %s\n", (
    npath + *it).c_str());
128                                         return false;
129                                 }
130
131                                 npath += *it + std::string("/");
132                         }
133
134                         return true;
135                 }
```

5.9.2.3 static std::string BCMFileIO::FileSystemUtil::FixDirectoryPath (const std::string & *dir*) [inline], [static]

dir をディレクトリ名として整形. 空の場合は"./"、文字列が入っている場合最後に "/"を追加

引数

in	*dir*	ディレクトリ名

戻り値

整形後のディレクトリ名

FileSystemUtil.h の 90 行目に定義があります。

参照元 BCMFileIO::BCMFileSaver::BCMFileSaver(), BCMFileIO::BCMFileLoader::LoadIndex(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation().

```
91                 {
92                         if( dir == std::string("") ){
93                                 return std::string("./");
94                         }else if( dir.rfind("/") != dir.length()-1 ){
95                                 return dir + std::string("/");
96                         }else{
97                                 return dir;
98                         }
99                 }
```

5.9.2.4 static std::string BCMFileIO::FileSystemUtil::GetDirectory (const std::string & *path*) [inline], [static]

path からディレクトリ名を抜き出す

引数

in	*path*	ファイルパス

戻り値

ディレクトリ名 (最後の"/"は残す)

FileSystemUtil.h の 57 行目に定義があります。

参照先 ConvertPath().

参照元 BCMFileIO::BCMFileLoader::BCMFileLoader(), BCMFileIO::BCMFileLoader::LoadAdditionalIndex(), BCMFileIO::BCMFileLoader::LoadIndex().

```
58                 {
59                         std::string cpath = ConvertPath(path);
60
61                         std::string dir;
62                         size_t p = cpath.rfind("/");
63                         if(p != std::string::npos)
64                         {
65                                 dir = cpath.substr(0, p+1);
66                         }else{
67                                 dir = std::string("./");
68                         }
69                         return dir;
70                 }
```

5.9.2.5 static std::string BCMFileIO::FileSystemUtil::GetFilePrefix (const std::string & *path*) [inline], [static]

path から拡張子以前のファイル名を抜き出す

引数

in	*path*	ファイルパス

戻り値

拡張子以前のファイル名

FileSystemUtil.h の 77 行目に定義があります。

参照先 ConvertPath().

```
78                 {
79                         std::string cpath = ConvertPath(path);
80                         std::string filename = cpath.substr(cpath.rfind("/")+1);
81
82                         return filename.substr(0, filename.rfind("."));
83                 }
```

5.9.2.6 static bool BCMFileIO::FileSystemUtil::split (const std::string & *input*, const char *delimiter*, std::vector< std::string > & *output*) [inline], [static]

文字列を指定の区切り文字で分割

引数

in	*input*	入力文字列
in	*delimiter*	区切り文字
out	*output*	分割された文字列リスト

戻り値

true: 分割成功のみ返却

FileSystemUtil.h の 144 行目に定義があります。

参照元 CreateDirectory().

```
145                 {
146                         output.clear();
147
148                         std::string istr = input;
149
150                         size_t p = 0;
151                         while( (p = istr.find(delimiter)) != std::string::npos ){
152                                 if(p != 0){
153                                         output.push_back( istr.substr(0, p) );
154                                 }
155                                 istr.erase(istr.begin(), istr.begin() + p + 1);
156                         }
157
158                         if( istr.length() != 0 ) output.push_back(istr);
159
160                         return true;
161                 }
```

このクラス詳解は次のファイルから抽出されました:

- FileSystemUtil.h

5.10 BCMFileIO::GridRleCode 構造体

RLE 圧縮符号の走査用構造体

`#include <BCMFileCommon.h>`

公開変数類

- bitVoxelCell c
  データ
- unsigned char len
  ラン長

5.10.1 詳解

RLE 圧縮符号の走査用構造体

BCMFileCommon.h の 87 行目に定義があります。

5.10.2 メンバ詳解

5.10.2.1 bitVoxelCell BCMFileIO::GridRleCode::c

データ

BCMFileCommon.h の 89 行目に定義があります。

5.10.2.2 unsigned char BCMFileIO::GridRleCode::len

ラン長

BCMFileCommon.h の 90 行目に定義があります。

この構造体詳解は次のファイルから抽出されました:

- BCMFileCommon.h

5.11 BCMFileIO::IdxBlock クラス

インデックスファイル用ブロック情報クラス

`#include <IdxBlock.h>`

BCMFileIO::IdxBlock 連携図

公開メンバ関数

- IdxBlock ()
  コンストラクタ

静的公開メンバ関数

- static IdxBlock * find (std::vector< IdxBlock > &idxBlockList, const int dataClassID)
- static IdxBlock * find (std::vector< IdxBlock > &idxBlockList, const std::string &name)
- static const IdxBlock * find (const std::vector< IdxBlock > &idxBlockList, const int dataClassID)
- static const IdxBlock * find (const std::vector< IdxBlock > &idxBlockList, const std::string &name)

公開変数類

- std::string rootDir
  インデックスファイルのディレクトリ
- std::string dataDir
  データディレクトリ
- std::vector< int > dataClassID
  データクラス*ID(***マルチコンポーネント対応のため配列***)*
- LB_DATA_TYPE dataType
  セルのデータ識別子
- std::string name
  系の名称
- LB_KIND kind
  リーフブロックタイプ
- unsigned int bitWidth
  セルあたりのビット幅
- unsigned int vc
  仮想セルサイズ
- std::string prefix
  ファイル名*Prefix*
- std::string extension
  ファイル拡張子
- bool isGather
  Gather **フラグ**
- bool isStepSubDir
  ステップごとのサブディレクトリフラグ
- IdxStep step
  タイムステップ情報
- bool separateVCUpdate

5.11.1 詳解

インデックスファイル用ブロック情報クラス

IdxBlock.h の 27 行目に定義があります。

5.11.2 構築子と解体子

5.11.2.1 BCMFileIO::IdxBlock::IdxBlock ( ) [inline]

コンストラクタ

IdxBlock.h の 32 行目に定義があります。

```
32                          :
33                      rootDir(std::string("")),
34                      dataDir(std::string("")),
35                      vc(0),
36                      isGather(false),
37                      isStepSubDir(false),
38                      separateVCUpdate(false)
39              {}
```

5.11.3 関数詳解

5.11.3.1 static IdxBlock* BCMFileIO::IdxBlock::find (std::vector< IdxBlock > & *idxBlockList,* const int *dataClassID*) [inline], [static]

データクラスID からブロック情報を取得するユーティリティ関数

引数

in	*idxBlockList*	ブロック情報リスト
in	*dataClassID*	データクラスID

戻り値

ブロック情報のポインタ

IdxBlock.h の 47 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::CreateLeafBlock(), BCMFileIO::BCMFileLoader::GetStep(), BCMFileIO::BCMFileLoader::LoadLeafBlock(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::BCMFileSaver::SaveLeafBlock().

```
47                                                                                  {
48                      for(std::vector<IdxBlock>::iterator it = idxBlockList.begin(); it != idxBlockList.
    end(); ++it){
49                              for(size_t i = 0; i < it->dataClassID.size(); i++){
50                                      if( it->dataClassID[i] == dataClassID){
51                                              return &(*it);
52                                      }
53                              }
54                      }
55                      return NULL;
56              }
```

5.11.3.2 static IdxBlock* BCMFileIO::IdxBlock::find (std::vector< IdxBlock > & *idxBlockList,* const std::string & *name*) [inline], [static]

系の名称からブロック情報を取得するユーティリティ関数

引数

in	*idxBlockList*	ブロック情報リスト
in	*name*	系の名称

戻り値

ブロック情報のポインタ

IdxBlock.h の 64 行目に定義があります。

```
64                                                                                                  {
65                          for(std::vector<IdxBlock>::iterator it = idxBlockList.begin(); it != idxBlockList.
    end(); ++it){
66                                  if( it->name == name){
67                                          return &(*it);
68                                  }
69                          }
70                          return NULL;
71                  }
```

5.11.3.3 static const IdxBlock* BCMFileIO::IdxBlock::find (const std::vector< IdxBlock > & *idxBlockList*, const int *dataClassID*) [inline], [static]

データクラスID からブロック情報を取得するユーティリティ関数 (const 用)

引数

in	*idxBlockList*	ブロック情報リスト
in	*dataClassID*	データクラスID

戻り値

ブロック情報のポインタ

IdxBlock.h の 79 行目に定義があります。

```
79
            {
80                          for(std::vector<IdxBlock>::const_iterator it = idxBlockList.begin(); it !=
    idxBlockList.end(); ++it){
81                                  for(size_t i = 0; i < it->dataClassID.size(); i++){
82                                          if( it->dataClassID[i] == dataClassID){
83                                                  return &(*it);
84                                          }
85                                  }
86                          }
87                          return NULL;
88                  }
```

5.11.3.4 static const IdxBlock* BCMFileIO::IdxBlock::find (const std::vector< IdxBlock > & *idxBlockList*, const std::string & *name*) [inline], [static]

系の名称からブロック情報を取得するユーティリティ関数 (const 用)

引数

in	*idxBlockList*	ブロック情報リスト
in	*name*	系の名称

戻り値

ブロック情報のポインタ

IdxBlock.h の 96 行目に定義があります。

```
96
            {
97                          for(std::vector<IdxBlock>::const_iterator it = idxBlockList.begin(); it !=
      idxBlockList.end(); ++it){
98                                  if( it->name == name){
99                                          return &(*it);
100                                  }
101                          }
102                          return NULL;
103                 }
```

5.11.4 メンバ詳解

5.11.4.1 unsigned int BCMFileIO::IdxBlock::bitWidth

セルあたりのビット幅

IdxBlock.h の 114 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::LeafBlockSaver::SaveCellID().

5.11.4.2 std::vector<int> BCMFileIO::IdxBlock::dataClassID

データクラスID(マルチコンポーネント対応のため配列)

IdxBlock.h の 110 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::BCMFileSaver::GetCellIDBlock(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation().

5.11.4.3 std::string BCMFileIO::IdxBlock::dataDir

データディレクトリ

IdxBlock.h の 108 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::LeafBlockSaver::SaveCellID().

5.11.4.4 LB_DATA_TYPE BCMFileIO::IdxBlock::dataType

セルのデータ識別子

IdxBlock.h の 111 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::LeafBlockSaver::SaveCellID(), BCMFileIO::LeafBlockSaver::SaveData().

5.11.4.5 std::string BCMFileIO::IdxBlock::extension

ファイル拡張子

IdxBlock.h の 117 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadCellID(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::LeafBlockSaver::SaveCellID().

5.11.4.6 bool BCMFileIO::IdxBlock::isGather

Gather フラグ

IdxBlock.h の 118 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::LeafBlockSaver::SaveCellID().

5.11.4.7 bool BCMFileIO::IdxBlock::isStepSubDir

ステップごとのサブディレクトリフラグ

IdxBlock.h の 119 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterDataInformation().

5.11.4.8 LB_KIND BCMFileIO::IdxBlock::kind

リーフブロックタイプ

IdxBlock.h の 113 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::LeafBlockSaver::SaveCellID().

5.11.4.9 std::string BCMFileIO::IdxBlock::name

系の名称

IdxBlock.h の 112 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation().

5.11.4.10 std::string BCMFileIO::IdxBlock::prefix

ファイル名Prefix

IdxBlock.h の 116 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadCellID(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::LeafBlockSaver::SaveCellID().

5.11.4.11 std::string BCMFileIO::IdxBlock::rootDir

インデックスファイルのディレクトリ

IdxBlock.h の 107 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndex(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::LeafBlockSaver::SaveCellID().

5.11.4.12 bool BCMFileIO::IdxBlock::separateVCUpdate

IdxBlock.h の 122 行目に定義があります。

5.11.4.13 IdxStep BCMFileIO::IdxBlock::step

タイムステップ情報

IdxBlock.h の 120 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::GetStep(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileSaver::RegisterDataInformation().

5.11.4.14 unsigned int BCMFileIO::IdxBlock::vc

仮想セルサイズ

IdxBlock.h の 115 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::BCMFileSaver::GetCellIDBlock(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::Load-

IndexData(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::LeafBlockSaver::SaveCellID().

このクラス詳解は次のファイルから抽出されました:

- IdxBlock.h

5.12 BCMFileIO::IdxProc 構造体

インデックスファイル用プロセス情報

`#include <BCMFileCommon.h>`

公開変数類

- std::string hostname
  ホスト名
- unsigned int rank
  ランク番号
- unsigned int rangeMin
  ブロック*ID* のレンジ最小値
- unsigned int rangeMax
  ブロック*ID* のレンジ最大値

5.12.1 詳解

インデックスファイル用プロセス情報

BCMFileCommon.h の 137 行目に定義があります。

5.12.2 メンバ詳解

5.12.2.1 std::string BCMFileIO::IdxProc::hostname

ホスト名

BCMFileCommon.h の 139 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadIndexProc().

5.12.2.2 unsigned int BCMFileIO::IdxProc::rangeMax

ブロックID のレンジ最大値

BCMFileCommon.h の 142 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadIndexProc().

5.12.2.3 unsigned int BCMFileIO::IdxProc::rangeMin

ブロックID のレンジ最小値

BCMFileCommon.h の 141 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadIndexProc().

5.12.2.4 unsigned int BCMFileIO::IdxProc::rank

ランク番号

BCMFileCommon.h の 140 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadIndexProc().

この構造体詳解は次のファイルから抽出されました:

- BCMFileCommon.h

5.13 BCMFileIO::IdxStep クラス

インデックスファイル用タイムステップ情報

`#include <IdxStep.h>`

公開メンバ関数

- IdxStep ()
  コンストラクタ
- IdxStep (const unsigned int rangeMin, const unsigned int rangeMax, const unsigned int rangeInterval=1)
- ~IdxStep ()
  デストラクタ
- bool SetRange (const unsigned int rangeMin, const unsigned int rangeMax, const unsigned int rangeInterval=1)
- void AddStep (const unsigned int step)
- void SubStep (const unsigned int step)
- bool IsCorrect (const unsigned int step) const
- void SetInitalTime (float time)
- void SetDeltaT (float deltaT)
- std::list< unsigned int > * GetStepList () const
- unsigned int GetRangeMin () const
- unsigned int GetRangeMax () const
- unsigned int GetRangeInterval () const
- const std::vector< unsigned int > & GetAddStepList () const
- const std::vector< unsigned int > & GetSubStepList () const
- float GetInitialTime () const
- float GetDeltaT () const

非公開変数類

- unsigned int m_rangeMin
  タイムステップレンジ *(Min)*
- unsigned int m_rangeMax
  タイムステップレンジ *(Max)*
- unsigned int m_rangeInterval
  タイムステップレンジ *(Interval)*
- std::vector< unsigned int > m_adds
  追加タイムステップリスト
- std::vector< unsigned int > m_subs
  削除タイムステップリスト
- float m_time
  Step = 0 **における時刻**
- float m_deltaT
  Step **間の時間幅**

5.13.1 詳解

インデックスファイル用タイムステップ情報

IdxStep.h の 28 行目に定義があります。

5.13.2 構築子と解体子

5.13.2.1 BCMFileIO::IdxStep::IdxStep ( )

コンストラクタ

IdxStep.cpp の 20 行目に定義があります。

```
20                          : m_rangeMin(0), m_rangeMax(0),
      m_rangeInterval(0), m_time(.0f), m_deltaT(.1f)
21         {
22         }
```

5.13.2.2 BCMFileIO::IdxStep::IdxStep (const unsigned int *rangeMin*, const unsigned int *rangeMax*, const unsigned int *rangeInterval* = 1)

コンストラクタ

引数

in	*rangeMin*	タイムステップレンジの開始インデックス
in	*rangeMax*	タイムステップレンジの終了インデックス
in	*rangeInterval*	ステップ間隔

IdxStep.cpp の 26 行目に定義があります。

参照先 SetRange().

```
27                  : m_rangeMin(0), m_rangeMax(0),
      m_rangeInterval(0), m_time(.0f), m_deltaT(.1f)
28          {
29                  SetRange(rangeMin, rangeMax, rangeInterval);
30          }
```

5.13.2.3 BCMFileIO::IdxStep::∼IdxStep ( )

デストラクタ

IdxStep.cpp の 34 行目に定義があります。

```
35          {
36          }
```

5.13.3 関数詳解

5.13.3.1 void BCMFileIO::IdxStep::AddStep (const unsigned int *step*)

追加ステップの設定

引数

in	*step*	追加ステップ

IdxStep.cpp の 52 行目に定義があります。

参照先 m_adds.

参照元 BCMFileIO::BCMFileLoader::LoadIndexStep().

```
53          {
54                  m_adds.push_back(step);
55          }
```

5.13.3.2 const std::vector< unsigned int > & BCMFileIO::IdxStep::GetAddStepList ( ) const

追加ステップリストを取得

戻り値

追加ステップリスト

IdxStep.cpp の 149 行目に定義があります。

参照先 m_adds.

```
150         {
151                 return m_adds;
152         }
```

5.13.3.3 float BCMFileIO::IdxStep::GetDeltaT ( ) const

Step 間の時間幅を取得

戻り値

Step 間の時間幅

IdxStep.cpp の 170 行目に定義があります。

参照先 m_deltaT.

```
171         {
172                 return m_deltaT;
173         }
```

5.13.3.4 float BCMFileIO::IdxStep::GetInitialTime ( ) const

Step = 0 における時刻を取得

戻り値

Step = 0 における時刻

IdxStep.cpp の 163 行目に定義があります。

参照先 m_time.

```
164         {
165                 return m_time;
166         }
```

5.13.3.5 unsigned int BCMFileIO::IdxStep::GetRangeInterval ( ) const

ステップ間隔を取得

戻り値

ステップ間隔

IdxStep.cpp の 142 行目に定義があります。

参照先 m_rangeInterval.

```
143         {
144                 return m_rangeInterval;
145         }
```

5.13.3.6 unsigned int BCMFileIO::IdxStep::GetRangeMax ( ) const

ステップの終了インデックスを取得

戻り値

終了インデックス

IdxStep.cpp の 135 行目に定義があります。

参照先 m_rangeMax.

```
136         {
137                 return m_rangeMax;
138         }
```

5.13.3.7 unsigned int BCMFileIO::IdxStep::GetRangeMin ( ) const

ステップの開始インデックスを取得

戻り値

開始インデックス

IdxStep.cpp の 128 行目に定義があります。

参照先 m_rangeMin.

```
129         {
130                 return m_rangeMin;
131         }
```

5.13.3.8 std::list< unsigned int > * BCMFileIO::IdxStep::GetStepList ( ) const

設定したStep のリストを取得

戻り値

ステップのリスト

覚え書き

リストはメソッド内で確保するため、リスト取得後、不要になったら解放してください。

IdxStep.cpp の 101 行目に定義があります。

参照先 m_adds, m_rangeInterval, m_rangeMax, m_rangeMin, m_subs.

```
102         {
103                 if( m_rangeInterval == 0 ){ return NULL; }
104
105                 std::list<unsigned int>*steps = new std::list<unsigned int>;
106
107                 for(unsigned int i = m_rangeMin; i <= m_rangeMax; i+=
    m_rangeInterval){
108                         steps->push_back(i);
109                 }
110
111                 // 追加リストからのステップ追加
112                 for(std::vector<unsigned int>::const_iterator it = m_adds.begin(); it !=
    m_adds.end(); ++it){
113                         steps->push_back((*it));
114                 }
115
116                 // 削除リストからステップ削除
```

```
117                 for(std::vector<unsigned int>::const_iterator it = m_subs.begin(); it !=
    m_subs.end(); ++it){
118                         steps->remove((*it));
119                 }
120
121                 steps->sort();
122
123                 return steps;
124         }
```

5.13.3.9 const std::vector< unsigned int > & BCMFileIO::IdxStep::GetSubStepList ( ) const

削除ステップリストを取得

戻り値

削除ステップリスト

IdxStep.cpp の 156 行目に定義があります。

参照先 m_subs.

```
157         {
158                 return m_subs;
159         }
```

5.13.3.10 bool BCMFileIO::IdxStep::IsCorrect (const unsigned int *step*) const

ステップが設定したリストに含まれるかを判定

引数

in	*step*	判定するステップ番号

戻り値

step がリストに含まれる場合 true, 含まれない場合 false

IdxStep.cpp の 66 行目に定義があります。

参照先 m_adds, m_rangeInterval, m_rangeMin, m_subs.

```
67         {
68                 for(std::vector<unsigned int>::const_iterator it = m_adds.begin(); it !=
    m_adds.end(); ++it){
69                         if( *it == step ){ return true; }
70                 }
71                 for(std::vector<unsigned int>::const_iterator it = m_subs.begin(); it !=
    m_subs.end(); ++it){
72                         if( *it == step ){ return false; }
73                 }
74
75                 if( m_rangeMin <= step && m_rangeMax >= step ){
76                         if( ((step - m_rangeMin) % m_rangeInterval) == 0 ){
77                                 return true;
78                         }else{
79                                 return false;
80                         }
81                 }
82                 return false;
83         }
```

5.13.3.11 void BCMFileIO::IdxStep::SetDeltaT (float *deltaT*)

Step 間の時刻幅を設定

引数

in	*deltaT*	Step 間の時刻幅

IdxStep.cpp の 94 行目に定義があります。

参照先 m_deltaT.

```
95         {
96                 m_deltaT = deltaT;
97         }
```

5.13.3.12 void BCMFileIO::IdxStep::SetInitalTime (float *time*)

Step = 0 における時刻を設定

引数

in	*time*	Step = 0 における時刻

IdxStep.cpp の 87 行目に定義があります。

参照先 m_time.

```
88         {
89                 m_time    = time;
90         }
```

5.13.3.13 bool BCMFileIO::IdxStep::SetRange (const unsigned int *rangeMin*, const unsigned int *rangeMax*, const unsigned int *rangeInterval* = 1)

タイムステップレンジ設定

引数

in	*rangeMin*	タイムステップレンジの開始インデックス
in	*rangeMax*	タイムステップレンジの終了インデックス
in	*rangeInterval*	ステップ間隔

戻り値

成功した場合 true, 失敗した場合 false

IdxStep.cpp の 40 行目に定義があります。

参照先 m_rangeInterval, m_rangeMax, m_rangeMin.

参照元 IdxStep(), BCMFileIO::BCMFileLoader::LoadIndexStep().

```
41         {
42                 if(rangeMax <= rangeMin){ return false; }
43                 m_rangeMin      = rangeMin;
44                 m_rangeMax      = rangeMax;
45                 m_rangeInterval = rangeInterval;
46
47                 return true;
48         }
```

5.13.3.14 void BCMFileIO::IdxStep::SubStep (const unsigned int *step*)

削除ステップの設定

引数

in	*step*	削除ステップ

IdxStep.cpp の 59 行目に定義があります。

参照先 m_subs.

参照元 BCMFileIO::BCMFileLoader::LoadIndexStep().

```
60          {
61                  m_subs.push_back(step);
62          }
```

5.13.4 メンバ詳解

5.13.4.1 std::vector<unsigned int> BCMFileIO::IdxStep::m_adds [private]

追加タイムステップリスト

IdxStep.h の 139 行目に定義があります。

参照元 AddStep(), GetAddStepList(), GetStepList(), IsCorrect().

5.13.4.2 float BCMFileIO::IdxStep::m_deltaT [private]

Step 間の時間幅

IdxStep.h の 142 行目に定義があります。

参照元 GetDeltaT(), SetDeltaT().

5.13.4.3 unsigned int BCMFileIO::IdxStep::m_rangeInterval [private]

タイムステップレンジ (Interval)

IdxStep.h の 138 行目に定義があります。

参照元 GetRangeInterval(), GetStepList(), IsCorrect(), SetRange().

5.13.4.4 unsigned int BCMFileIO::IdxStep::m_rangeMax [private]

タイムステップレンジ (Max)

IdxStep.h の 137 行目に定義があります。

参照元 GetRangeMax(), GetStepList(), SetRange().

5.13.4.5 unsigned int BCMFileIO::IdxStep::m_rangeMin [private]

タイムステップレンジ (Min)

IdxStep.h の 136 行目に定義があります。

参照元 GetRangeMin(), GetStepList(), IsCorrect(), SetRange().

5.13.4.6 std::vector<unsigned int> BCMFileIO::IdxStep::m_subs [private]

削除タイムステップリスト

IdxStep.h の 140 行目に定義があります。

参照元 GetStepList(), GetSubStepList(), IsCorrect(), SubStep().

5.13.4.7 float BCMFileIO::IdxStep::m_time [private]

Step = 0 における時刻

IdxStep.h の 141 行目に定義があります。

参照元 GetInitialTime(), SetInitalTime().

このクラス詳解は次のファイルから抽出されました:

- IdxStep.h
- IdxStep.cpp

5.14 BCMFileIO::IdxUnit 構造体

インデックスファイル用単位系情報

`#include <BCMFileCommon.h>`

公開変数類

- std::string length
  長さ単位 *(NonDimensional, m, cm, mm)*
- double L0_scale
  規格化に用いたスケール *(**単位**.**指定単位**)*
- std::string velocity
  時間単位 *(NonDimensional, Dimensional)*
- double V0_scale
  規格化に用いた時間スケール *(**単位**:Dimensional **の場合** m/s)*

5.14.1 詳解

インデックスファイル用単位系情報

BCMFileCommon.h の 128 行目に定義があります。

5.14.2 メンバ詳解

5.14.2.1 double BCMFileIO::IdxUnit::L0_scale

規格化に用いたスケール (単位:指定単位)

BCMFileCommon.h の 131 行目に定義があります。

参照元 BCMFileIO::BCMFileSaver::BCMFileSaver(), BCMFileIO::BCMFileLoader::LoadIndex(), BCMFileIO::BCMFileSaver::SaveIndexCellID(), BCMFileIO::BCMFileSaver::SaveIndexData().

5.14.2.2 std::string BCMFileIO::IdxUnit::length

長さ単位 (NonDimensional, m, cm, mm)

BCMFileCommon.h の 130 行目に定義があります。

参照元 BCMFileIO::BCMFileSaver::BCMFileSaver(), BCMFileIO::BCMFileLoader::LoadIndex(), BCMFileIO::BCMFileSaver::SaveIndexCellID(), BCMFileIO::BCMFileSaver::SaveIndexData().

5.14.2.3 double BCMFileIO::IdxUnit::V0_scale

規格化に用いた時間スケール (単位:Dimensional の場合 m/s)

BCMFileCommon.h の 133 行目に定義があります。

参照元 BCMFileIO::BCMFileSaver::BCMFileSaver(), BCMFileIO::BCMFileLoader::LoadIndex(), BCMFileIO::BCMFileSaver::SaveIndexData().

5.14.2.4 std::string BCMFileIO::IdxUnit::velocity

時間単位 (NonDimensional, Dimensional)

BCMFileCommon.h の 132 行目に定義があります。

参照元 BCMFileIO::BCMFileSaver::BCMFileSaver(), BCMFileIO::BCMFileLoader::LoadIndex(), BCMFileIO::BCMFileSaver::SaveIndexData().

この構造体詳解は次のファイルから抽出されました:

- BCMFileCommon.h

5.15 BCMFileIO::LBCellIDHeader 構造体

LeafBlock のCellID ヘッダ構造体

`#include <BCMFileCommon.h>`

公開変数類

- uint64_t numBlock
  ブロック数
- uint64_t compSize
  圧縮符号サイズ (バイト単位)

5.15.1 詳解

LeafBlock のCellID ヘッダ構造体

BCMFileCommon.h の 77 行目に定義があります。

5.15.2 メンバ詳解

5.15.2.1 uint64_t BCMFileIO::LBCellIDHeader::compSize

圧縮符号サイズ (バイト単位)

BCMFileCommon.h の 80 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::DecompCellIDData(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadCellIDData(), BCMFileIO::LeafBlockLoader::LoadCellIDHeader(), BCMFileIO::-LeafBlockSaver::SaveCellID().

5.15.2.2 uint64_t BCMFileIO::LBCellIDHeader::numBlock

ブロック数

BCMFileCommon.h の 79 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::DecompCellIDData(), BCMFileIO::LeafBlockLoader::LoadCellIDData(), BCMFileIO::LeafBlockLoader::LoadCellIDHeader(), BCMFileIO::LeafBlockSaver::SaveCellID().

この構造体詳解は次のファイルから抽出されました:

- BCMFileCommon.h

5.16 BCMFileIO::LBHeader 構造体

LeafBlock ファイルヘッダ構造体

`#include <BCMFileCommon.h>`

公開変数類

- unsigned int identifier
  エンディアン識別子
- unsigned char kind
  ブロックファイル種類
- unsigned char dataType
  1 **セルあたりのサイズ**
- unsigned short bitWidth
  1 **セルあたりのビット幅**
- unsigned int vc
  仮想セルサイズ
- unsigned int size [3]

ブロックサイズ

- uint64_t numBlock

  ファイルに記載されている総ブロック数

5.16.1 詳解

LeafBlock ファイルヘッダ構造体

BCMFileCommon.h の 64 行目に定義があります。

5.16.2 メンバ詳解

5.16.2.1 unsigned short BCMFileIO::LBHeader::bitWidth

1 セルあたりのビット幅

BCMFileCommon.h の 69 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::DecompCellIDData(), BCMFileIO::LeafBlockLoader::GetBitVoxelSize(), BCMFileIO::LeafBlockLoader::LoadCellID(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadHeader().

5.16.2.2 unsigned char BCMFileIO::LBHeader::dataType

1 セルあたりのサイズ

BCMFileCommon.h の 68 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::Load_BlockContents(), BCMFileIO::LeafBlockLoader::LoadData().

5.16.2.3 unsigned int BCMFileIO::LBHeader::identifier

エンディアン識別子

BCMFileCommon.h の 66 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadHeader(), BCMFileIO::LeafBlockSaver::SaveCellID().

5.16.2.4 unsigned char BCMFileIO::LBHeader::kind

ブロックファイル種類

BCMFileCommon.h の 67 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::LoadCellID(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadData().

5.16.2.5 uint64_t BCMFileIO::LBHeader::numBlock

ファイルに記載されている総ブロック数

BCMFileCommon.h の 72 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadHeader().

5.16.2.6 unsigned int BCMFileIO::LBHeader::size[3]

ブロックサイズ

BCMFileCommon.h の 71 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::DecompCellIDData(), BCMFileIO::LeafBlockLoader::GetBitVoxelSize(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::LeafBlockLoader::LoadHeader().

5.16.2.7 unsigned int BCMFileIO::LBHeader::vc

仮想セルサイズ

BCMFileCommon.h の 70 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::DecompCellIDData(), BCMFileIO::LeafBlockLoader::GetBitVoxelSize(), BCMFileIO::LeafBlockLoader::Load_BlockContents(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::LeafBlockLoader::LoadHeader().

この構造体詳解は次のファイルから抽出されました:

- BCMFileCommon.h

5.17 BCMFileIO::LeafBlockLoader クラス

LeafBlock ファイルを読み込むクラス

`#include <LeafBlockLoader.h>`

クラス

- struct CellIDCapsule
  グリッドヘッダとデータを一括りにした構造体

静的公開メンバ関数

- static bool LoadCellID (const std::string &dir, const IdxBlock *ib, const MPI::Intracomm &comm, PartitionMapper *pmapper, LBHeader &header, std::vector< CellIDCapsule > &cidCapsules)
- static bool LoadCellID_Gather (const std::string &dir, const IdxBlock *ib, const MPI::Intracomm &comm, PartitionMapper *pmapper, LBHeader &header, std::vector< CellIDCapsule > &cidCapsules)
- static unsigned char * DecompCellIDData (const LBHeader &header, const CellIDCapsule &cidCapsule)
- static bool LoadData (const MPI::Intracomm &comm, const IdxBlock *ib, BlockManager &blockManager, PartitionMapper *pmapper, const int vc, const unsigned int step)

- template<typename T >
  static bool CopyBufferToScalar3D (BlockManager &blockManager, const int dataClassID, const int blockID, const int vc, const T *buf)

静的非公開メンバ関数

- static size_t GetBitVoxelSize (const LBHeader &hdr, size_t numBlocks)
  BitVoxel **サイズを取得する**
- static bool LoadHeader (FILE *fp, LBHeader &hdr, bool &isNeedSwap)
  ヘッダを読み込む
- static bool LoadCellIDHeader (FILE *fp, LBCellIDHeader &chdr, const bool isNeedSwap)
  CellID **ヘッダを読み込む**
- static bool LoadCellIDData (FILE *fp, unsigned char **data, const LBHeader &hdr, const LBCellIDHeader &chdr, const bool isNeedSwap)
  CellID **データを読み込む**
- static unsigned char * Load_BlockContents (FILE *fp, const LBHeader &hdr, const Vec3i &bsz, const int vc, const bool isNeedSwap)
  ブロックコンテンツを読み込む

5.17.1 詳解

LeafBlock ファイルを読み込むクラス

LeafBlockLoader.h の 31 行目に定義があります。

5.17.2 関数詳解

5.17.2.1 template<typename T > bool BCMFileIO::LeafBlockLoader::CopyBufferToScalar3D (BlockManager & *blockManager*, const int *dataClassID*, const int *blockID*, const int *vc*, const T * *buf*) [static]

データバッファからBlockManager 配下のBlock にデータをコピー

引数

in	*blockManager*	ブロックマネージャ
in	*dataClassID*	データクラスID
in	*blockID*	ブロックID (プロセス内部でのブロック番号)
in	*vc*	仮想セルサイズ
in	*buf*	コピー元データバッファ

戻り値

成功した場合 true, 失敗した場合 false

LeafBlockLoader.cpp の 528 行目に定義があります。

参照先 Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z.

参照元 BCMFileIO::BCMFileLoader::LoadLeafBlock().

```
529         {
530                 Vec3i size = blockManager.getSize();
531
532                 BlockBase* block = blockManager.getBlock(blockID);
533                 Scalar3D<T>* mesh = dynamic_cast< Scalar3D<T>* >(block->getDataClass(dataClassID));
```

```
534                 T* data      = mesh->getData();
535                 Index3DS idx = mesh->getIndex();
536 
537                 for(int z = -vc; z < size.z + vc; z++){
538                         for(int y = -vc; y < size.y + vc; y++){
539                                 for(int x = -vc; x < size.x + vc; x++){
540                                         size_t loc = ( (x+vc) + ((y+vc) + (z+vc) * (size.
    y + (vc*2))) * (size.x + (vc*2)) );
541                                         data[idx(x, y, z)] = buf[loc];
542                                 }
543                         }
544                 }
545                 return true;
546         }
```

5.17.2.2 unsigned char * BCMFileIO::LeafBlockLoader::DecompCellIDData (const LBHeader & *header*, const CellIDCapsule & *cidCapsule*) [static]

圧縮データの展開

引数

in	*header*	LeafBlock ファイルヘッダ
in	*cidCapsule*	CellID カプセル

戻り値

展開後のデータ (失敗した場合NULL を返す．)

覚え書き

cidCapsule の data はこの関数内で解放される．

LeafBlockLoader.cpp の 358 行目に定義があります。

参照先 BCMFileIO::LBHeader::bitWidth, BCMFileIO::LBCellIDHeader::compSize, BCMFileIO::LeafBlockLoader::CellIDCapsule::data, BCMFileIO::LeafBlockLoader::CellIDCapsule::header, BCMFileIO::LBCellIDHeader::numBlock, BCMFileIO::LBHeader::size, BCMFileIO::LBHeader::vc.

参照元 BCMFileIO::BCMFileLoader::LoadLeafBlock().

```
359         {
360                 unsigned char* ret = NULL;
361 
362                 size_t blockSize = (header.size[0] + header.vc*2) * (header.size[1] + header.vc*2) * (
    header.size[2] + header.vc*2);
363                 size_t dataSize  = blockSize * cc.header.numBlock;
364 
365                 bitVoxelCell* bitVoxel = NULL;
366                 if( cc.header.compSize != 0){
367                         size_t bitVoxelSize = BitVoxel::GetSize(dataSize, header.bitWidth)
    ;
368                         size_t dsize = bitVoxelSize * sizeof(bitVoxelCell);
369                         bitVoxel = BCMRLE::Decode<bitVoxelCell, unsigned char>(cc.data, cc.header.compSize,
     dsize);
370                         delete [] cc.data;
371                 }else{
372                         bitVoxel = reinterpret_cast<bitVoxelCell*>(cc.data);
373                 }
374 
375                 ret = BitVoxel::Decompress(dataSize, bitVoxel, header.bitWidth);
376 
377                 delete [] bitVoxel;
378 
379                 return ret;
380         }
```

5.17.2.3 size_t BCMFileIO::LeafBlockLoader::GetBitVoxelSize (const LBHeader & *hdr*, size_t *numBlocks*) [inline], [static], [private]

BitVoxel サイズを取得する

LeafBlockLoader.cpp の 34 行目に定義があります。

参照先 BCMFileIO::LBHeader::bitWidth, BCMFileIO::LBHeader::size, BCMFileIO::LBHeader::vc.

```
34                                                                                  {
35                 size_t blockSize = (hdr.size[0] + hdr.vc * 2) * (hdr.size[1] + hdr.vc * 2) * (hdr.size[2] +
       hdr.vc * 2);
36                 return BitVoxel::GetSize(blockSize * numBlocks, hdr.bitWidth);
37         }
```

5.17.2.4 unsigned char * BCMFileIO::LeafBlockLoader::Load_BlockContents (FILE * *fp*, const LBHeader & *hdr*, const Vec3i & *bsz*, const int *vc*, const bool *isNeedSwap*) [static], [private]

ブロックコンテンツを読み込む

LeafBlockLoader.cpp の 384 行目に定義があります。

参照先 BCMFileIO::BSwap16(), BCMFileIO::BSwap32(), BCMFileIO::BSwap64(), BCMFileIO::LBHeader::dataType, BCMFileIO::DUMMY(), BCMFileIO::LBHeader::vc, Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z.

```
385         {
386                 const static size_t typeByteTable[10] = { 1, 1, 2, 2, 4, 4, 8, 8, 4, 8 };
387                 static void (*BSwap[10])(void*) = { DUMMY, DUMMY,
       BSwap16, BSwap16, BSwap32, BSwap32, BSwap64,
       BSwap64, BSwap32, BSwap64 };
388
389                 size_t typeByte = typeByteTable[hdr.dataType];
390
391                 Vec3i ibsz( bsz.x + vc*2,     bsz.y + vc*2,     bsz.z + vc*2);
392                 Vec3i fbsz( bsz.x + hdr.vc*2, bsz.y + hdr.vc*2, bsz.z + hdr.vc*2);
393
394                 unsigned char* block = new unsigned char[ibsz.x * ibsz.y * ibsz.z * typeByte];
395                 unsigned char* buf   = new unsigned char[fbsz.x * fbsz.y * fbsz.z * typeByte];
396
397                 memset(block, 0, sizeof(unsigned char) * ibsz.x * ibsz.y * ibsz.z * typeByte);
398
399                 fread(buf, sizeof(unsigned char), fbsz.x * fbsz.y * fbsz.z * typeByte, fp);
400
401                 if( isNeedSwap ){
402                         for(int i = 0; i < fbsz.x * fbsz.y * fbsz.z; i++){
403                                 BSwap[hdr.dataType](&buf[i * typeByte]);
404                         }
405                 }
406
407                 if( vc > hdr.vc ){
408                         unsigned int vcd = vc - hdr.vc;
409                         for(int z = 0; z < fbsz.z; z++){
410                                 for(int y = 0; y < fbsz.y; y++){
411                                         size_t ibloc = 0 + vcd + ( (y + vcd) + (z + vcd) * ibsz.y ) * ibsz.
       x;
412                                         size_t fbloc = 0 +     + (  y        +  z        * fbsz.y ) * fbsz.
       x;
413                                         memcpy(&block[ibloc * typeByte], &buf[fbloc * typeByte], typeByte *
        fbsz.x );
414                                 }
415                         }
416                 }else{
417                         unsigned int vcd = hdr.vc - vc;
418                         for(int z = 0; z < ibsz.z; z++){
419                                 for(int y = 0; y < ibsz.y; y++){
420                                         size_t ibloc = 0 +     + (  y        +  z        * ibsz.y ) * ibsz.
       x;
421                                         size_t fbloc = 0 + vcd + ( (y + vcd) + (z + vcd) * fbsz.y ) * fbsz.
       x;
422                                         memcpy(&block[ibloc * typeByte], &buf[fbloc * typeByte], typeByte *
        ibsz.x );
423                                 }
424                         }
425                 }
```

```
426
427                 delete [] buf;
428                 return block;
429         }
```

5.17.2.5 bool BCMFileIO::LeafBlockLoader::LoadCellID (const std::string & *dir*, const IdxBlock * *ib*, const MPI::Intracomm & *comm*, PartitionMapper * *pmapper*, LBHeader & *header*, std::vector< CellIDCapsule > & *cidCapsules*) [static]

LeafBlock ファイル (CellID) の読み込み (Gather なし)

引数

in	*dir*	入力元ディレクトリ
in	*ib*	ブロック情報
in	*comm*	MPI コミュニケータ
in	*pmapper*	MxN データマッパ
out	*header*	LeafBlock ファイルヘッダ
out	*cidCapsules*	自プロセスが担当するデータのリスト

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

cidCapsules に入る値はファイルから読み込んだ生の状態． 圧縮符号や bitVoxel の展開，真に必要なデータの選択は ここでは実施しない

LeafBlockLoader.cpp の 111 行目に定義があります。

参照先 BCMFileIO::LBHeader::bitWidth, BCMFileIO::LeafBlockLoader::CellIDCapsule::data, BCMFileIO::IdxBlock::extension, BCMFileIO::PartitionMapper::GetFDIDLists(), BCMFileIO::LeafBlockLoader::CellIDCapsule::header, BCMFileIO::LBHeader::kind, BCMFileIO::LB_CELLID, BCMFileIO::IdxBlock::prefix.

参照元 BCMFileIO::BCMFileLoader::LoadLeafBlock().

```
117         {
118                 using namespace std;
119
120                 cidCapsules.clear();
121
122                 vector<PartitionMapper::FDIDList> fdidlists;
123                 pmapper->GetFDIDLists(comm.Get_rank(), fdidlists);
124
125                 cidCapsules.reserve(fdidlists.size());
126
127                 bool err = false;
128
129                 for(vector<PartitionMapper::FDIDList>::const_iterator file = fdidlists.begin(); file !=
    fdidlists.end(); ++file){
130                         char filename[128];
131                         sprintf(filename, "%s_%06d.%s", ib->prefix.c_str(), file->FID, ib->extension.c_str(
    ));
132                         string filepath = dir + string(filename);
133
134                         FILE *fp = NULL;
135                         if( (fp = fopen(filepath.c_str(), "rb")) == NULL ){
136                                 Logger::Error("file open error \"%s\" [%s:%d]\n", filepath.
    c_str(), __FILE__, __LINE__);
137                                 err = true;
138                                 break;
139                         }
140
141                         bool isNeedSwap = false;
142
143                         LBHeader hdr;
```

```
144
145                         if( !LoadHeader(fp, hdr, isNeedSwap) ){
146                                 Logger::Error("%s is not leafBlock file [%s:%d]\n", filepath.
    c_str(), __FILE__, __LINE__);
147                                 fclose(fp); err = true; break;
148                         }
149
150                         if(hdr.kind != static_cast<unsigned char>(LB_CELLID)){
151                                 Logger::Error("%s is not CellID file [%s:%d]\n", filepath.
    c_str(), __FILE__, __LINE__);
152                                 fclose(fp); err = true; break;
153                         }
154                         if(hdr.bitWidth < 1 && header.bitWidth > 5) {
155                                 Logger::Error("%s is not CEllID file [%s:%d]\n", filepath.
    c_str(), __FILE__, __LINE__);
156                                 fclose(fp); err = true; break;
157                         }
158
159                         CellIDCapsule cc;
160                         if( !LoadCellIDHeader(fp, cc.header, isNeedSwap) ){
161                                 fclose(fp); err = true; break;
162                         }
163
164                         LoadCellIDData(fp, &cc.data, hdr, cc.header, isNeedSwap);
165
166                         cidCapsules.push_back(cc);
167                         header = hdr;
168                         fclose(fp);
169                 }
170
171                 if( err ){
172                         Logger::Error("Clear cidCapsules [%s:%d]\n", __FILE__, __LINE__);
173                         // データロードに失敗したためロード済みのデータを破棄
174                         for(vector<CellIDCapsule>::iterator it = cidCapsules.begin(); it != cidCapsules.end
    (); ++it){
175                                 if( it->data != NULL) delete [] it->data;
176                         }
177                         cidCapsules.clear();
178                         return false;
179                 }
180
181                 return true;
182         }
```

5.17.2.6 bool BCMFileIO::LeafBlockLoader::LoadCellID_Gather (const std::string & *dir*, const IdxBlock * *ib*, const MPI::Intracomm & *comm*, PartitionMapper * *pmapper*, LBHeader & *header*, std::vector< CellIDCapsule > & *cidCapsules*) [static]

LeafBlock ファイル (CellID) の読み込み (Gather あり)

引数

in	*dir*	入力元ディレクトリ
in	*ib*	ブロック情報
in	*comm*	MPI コミュニケータ
in	*pmapper*	MxN データマッパ
out	*header*	LeafBlock ファイルヘッダ
out	*cidCapsules*	自プロセスが担当するデータのリスト

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

cidCapsules に入る値はファイルから読み込んだ生の状態． 圧縮符号や bitVoxel の展開，真に必要なデータの選択は ここでは実施しない

LeafBlockLoader.cpp の 184 行目に定義があります。

参照先 BCMFileIO::LBHeader::bitWidth, BCMFileIO::BSwap64(), BCMFileIO::LBCellIDHeader::compSize, BCMFileIO::LeafBlockLoader::CellIDCapsule::data, BCMFileIO::IdxBlock::extension, BCMFileIO::PartitionMapper::GetFDIDLists(), BCMFileIO::PartitionMapper::GetWriteProcs(), BCMFileIO::LeafBlockLoader::CellIDCapsule::header, BCMFileIO::LBHeader::kind, BCMFileIO::LB_CELLID, BCMFileIO::LBHeader::numBlock, BCMFileIO::IdxBlock::prefix.

参照元 BCMFileIO::BCMFileLoader::LoadLeafBlock().

```
190         {
191                 using namespace std;
192 
193                 cidCapsules.clear();
194 
195                 int rank = comm.Get_rank();
196 
197                 // rank 0 でファイルをロード
198                 if( rank == 0 ){
199                         // ファイルロード
200                         char filename[128];
201                         sprintf(filename, "%s.%s", ib->prefix.c_str(), ib->extension.c_str());
202                         string filepath = dir + string(filename);
203 
204                         FILE *fp = NULL;
205                         if( (fp = fopen(filepath.c_str(), "rb")) == NULL ){
206                                 printf("err : file open error \"%s\" [%s:%d]\n", filepath.c_str(), __FILE__
    , __LINE__);
207 
208                                 // ファイルロードエラーを全プロセスに通知
209                                 unsigned char loadError = 1; comm.Bcast(&loadError, 1, MPI::CHAR, 0);
210                                 return false;
211                         }
212 
213                         bool isNeedSwap = false;
214 
215                         LBHeader hdr;
216 
217                         if( !LoadHeader(fp, hdr, isNeedSwap) ){
218                                 Logger::Error("%s is not leafBlock file [%s:%d]\n", filepath.
    c_str(), __FILE__, __LINE__);
219                                 // ファイルロードエラーを全プロセスに通知
220                                 unsigned char loadError = 1; comm.Bcast(&loadError, 1, MPI::CHAR, 0);
221                                 fclose(fp); return false;
222                         }
223 
224                         if(hdr.kind != static_cast<unsigned char>(LB_CELLID)){
225                                 Logger::Error("%s is not Grid file [%s:%d]\n", filepath.c_str(
    ), __FILE__, __LINE__);
226                                 // ファイルロードエラーを全プロセスに通知
227                                 unsigned char loadError = 1; comm.Bcast(&loadError, 1, MPI::CHAR, 0);
228                                 fclose(fp); return false;
229                         }
230 
231                         if(hdr.bitWidth < 1 && header.bitWidth > 5) {
232                                 Logger::Error("%s is not Grid file [%s:%d]\n", filepath.c_str(
    ), __FILE__, __LINE__);
233                                 // ファイルロードエラーを全プロセスに通知
234                                 unsigned char loadError = 1; comm.Bcast(&loadError, 1, MPI::CHAR, 0);
235                                 fclose(fp); return false;
236                         }
237 
238                         header = hdr;
239 
240                         uint64_t wnp = 0;
241                         fread(&wnp, sizeof(uint64_t), 1, fp);
242                         if( isNeedSwap ){
243                                 BSwap64(&wnp);
244                         }
245 
246                         if( wnp != pmapper->GetWriteProcs() ){
247                                 printf("err : %s's write procs is invalid  [%s:%d]\n", filepath.c_str(),
    __FILE__, __LINE__);
248                                 // ファイルロードエラーを全プロセスに通知
249                                 unsigned char loadError = 1; comm.Bcast(&loadError, 1, MPI::CHAR, 0);
250                                 fclose(fp); return false;
251                         }
252 
253                         vector<LBCellIDHeader> chs(wnp);
254                         for(int i = 0; i < wnp; i++){
255                                 if( !LoadCellIDHeader(fp, chs[i], isNeedSwap) ){
256                                         // ファイルロードエラーを全プロセスに通知
257                                         unsigned char loadError = 1; comm.Bcast(&loadError, 1, MPI::CHAR, 0
    );
258                                         fclose(fp); return false;
259                                 }
260                         }
```

```
261
262                         vector<unsigned char*> contents(wnp);
263
264                         for(int i = 0; i < wnp; i++){
265                                 contents[i] = NULL;
266                                 LoadCellIDData( fp, &contents[i], hdr, chs[i], isNeedSwap );
267                         }
268
269                         fclose(fp);
270
271                         // ファイルロード完了を全プロセスに通知
272                         unsigned char loadError = 0; comm.Bcast(&loadError, 1, MPI::CHAR, 0);
273
274                         // ヘッダ情報をブロードキャスト
275                         comm.Bcast(&header, sizeof(LBHeader), MPI::CHAR, 0);
276                         // Grid ヘッダ情報をブロードキャスト
277                         comm.Bcast(&chs[0], wnp * sizeof(LBCellIDHeader), MPI::CHAR, 0);
278
279                         // 各計算ノードにデータを送信
280                         for(int i = 1; i < comm.Get_size(); i++){
281                                 vector<PartitionMapper::FDIDList> fdidlists;
282                                 pmapper->GetFDIDLists(i, fdidlists);
283                                 for(vector<PartitionMapper::FDIDList>::iterator file = fdidlists.begin();
    file != fdidlists.end(); ++file){
284                                         size_t sz = 0;
285                                         if( chs[file->FID].compSize == 0){
286                                                 sz = GetBitVoxelSize(header, chs[file->FID].
    numBlock) * sizeof(bitVoxelCell);
287                                         }else{
288                                                 sz = chs[file->FID].compSize;
289                                         }
290
291                                         // 送信
292                                         int tag = file->FID;
293                                         comm.Send(contents[file->FID], sz, MPI::CHAR, i, tag);
294                                 }
295                         }
296
297                         // Rank 0 用のデータコピー
298
299                         bool freeMask[wnp];
300                         for( int i = 0; i < wnp; i++){ freeMask[i] = true; }
301
302                         vector<PartitionMapper::FDIDList> fdidlists;
303                         pmapper->GetFDIDLists(rank, fdidlists);
304                         for(vector<PartitionMapper::FDIDList>::iterator file = fdidlists.begin(); file !=
    fdidlists.end(); ++file){
305                                 CellIDCapsule cc;
306                                 cc.header = chs[file->FID];
307                                 cc.data   = contents[file->FID];
308                                 cidCapsules.push_back(cc);
309                                 freeMask[file->FID] = false;
310                         }
311
312                         // 不要なデータを解放
313                         for(int i = 0; i < wnp; i++){
314                                 if(freeMask[i]) delete [] contents[i];
315                         }
316                 }
317                 // rank 0 以外
318                 else
319                 {
320                         // rank 0 からの通知を受け取る
321                         unsigned char loadError = 0; comm.Bcast(&loadError, 1, MPI::CHAR, 0);
322                         if(loadError == 1){
323                                 return false;
324                         }
325
326                         // ヘッダ情報を取得
327                         comm.Bcast(&header, sizeof(LBHeader), MPI::CHAR, 0);
328
329                         // Grid ヘッダ情報を取得
330                         int wnp = pmapper->GetWriteProcs();
331                         vector<LBCellIDHeader> chs(wnp);
332                         comm.Bcast(&chs[0], wnp * sizeof(LBCellIDHeader), MPI::CHAR, 0);
333
334                         vector<PartitionMapper::FDIDList> fdidlists;
335                         pmapper->GetFDIDLists(rank, fdidlists);
336                         for(vector<PartitionMapper::FDIDList>::iterator file = fdidlists.begin(); file !=
    fdidlists.end(); ++file){
337                                 CellIDCapsule cc;
338                                 cc.header = chs[file->FID];
339                                 size_t sz = 0;
340                                 if( chs[file->FID].compSize == 0){
341                                         sz = GetBitVoxelSize(header, chs[file->FID].numBlock
    ) * sizeof(bitVoxelCell);
342                                 }else{
```

```
343                                         sz = chs[file->FID].compSize;
344                                 }
345                                 cc.data = new unsigned char[sz];
346
347                                 // 受信
348                                 int tag = file->FID;
349                                 comm.Recv(cc.data, sz, MPI::CHAR, 0, tag);
350
351                                 cidCapsules.push_back(cc);
352                         }
353                 }
354                 return true;
355         }
```

5.17.2.7 bool BCMFileIO::LeafBlockLoader::LoadCellIDData (FILE * *fp*, unsigned char ** *data*, const LBHeader & *hdr*, const LBCellIDHeader & *chdr*, const bool *isNeedSwap*) [inline], [static], [private]

CellID データを読み込む

LeafBlockLoader.cpp の 80 行目に定義があります。

参照先 BCMFileIO::BSwap32(), BCMFileIO::LBCellIDHeader::compSize, BCMFileIO::LBCellIDHeader::numBlock.

```
81         {
82                 size_t sz = 0;
83                 if( chdr.compSize == 0 ){
84                         sz = GetBitVoxelSize(hdr, chdr.numBlock) * sizeof(
       bitVoxelCell);
85                 }else{
86                         sz = chdr.compSize;
87                 }
88
89                 *data = new unsigned char[sz];
90
91                 fread(*data, sizeof(unsigned char), sz, fp);
92
93                 if( isNeedSwap ){
94                         if( chdr.compSize == 0 ){
95                                 size_t bitVoxelSize = GetBitVoxelSize(hdr, chdr.numBlock);
96                                 bitVoxelCell* bitVoxel = reinterpret_cast<
       bitVoxelCell*>(*data);
97                                 for(int i = 0; i < bitVoxelSize; i++){
98                                         BSwap32(&bitVoxel[i]);
99                                 }
100                         }else{
101                                 GridRleCode* p = reinterpret_cast<GridRleCode*>(*data);
102                                 for(int i = 0; i < sz / sizeof(GridRleCode); i++){
103                                         BSwap32(&(p[i].c));
104                                 }
105                         }
106                 }
107
108                 return true;
109         }
```

5.17.2.8 bool BCMFileIO::LeafBlockLoader::LoadCellIDHeader (FILE * *fp*, LBCellIDHeader & *chdr*, const bool *isNeedSwap*) [inline], [static], [private]

CellID ヘッダを読み込む

LeafBlockLoader.cpp の 66 行目に定義があります。

参照先 BCMFileIO::BSwap64(), BCMFileIO::LBCellIDHeader::compSize, BCMFileIO::LBCellIDHeader::numBlock.

```
67         {
68                 fread(&chdr, sizeof(LBCellIDHeader), 1, fp);
69                 if( isNeedSwap ){
70                         BSwap64(&chdr.numBlock);
71                         BSwap64(&chdr.compSize);
72                 }
```

```
73                 if(chdr.compSize != 0 && (chdr.compSize % sizeof(GridRleCode)) != 0){
74                         Logger::Error("compress size is invalid\n");
75                         return false;
76                 }
77                 return true;
78         }
```

5.17.2.9 bool BCMFileIO::LeafBlockLoader::LoadData (const MPI::Intracomm & *comm*, const IdxBlock * *ib*, BlockManager & *blockManager*, PartitionMapper * *pmapper*, const int *vc*, const unsigned int *step*) [static]

LeafBlock ファイル (Scalar) の読み込み

引数

in	*comm*	MPI コミュニケータ
in	*ib*	ブロック情報
in	*blockManager*	ブロックマネージャ
in	*pmapper*	MxN データマッパ
in	*vc*	内部構造の仮想セルサイズ
in	*step*	読み込むデータのタイムステップインデックス番号

戻り値

成功した場合 true, 失敗した場合 false

覚え書き

このメソッドの実行により、LeafBlock ファイルの中身がBlockManager 配下のBlock に格納されます。仮想セルサイズは、ファイルに記載されている仮想セルサイズと関係なく指定できます。 ブロック間の仮想セルの同期は行っていないため、ファイルの仮想セルサイズよりも大きい値を入れた 場合、読めない値は 0 で埋まります。

LeafBlockLoader.cpp の 431 行目に定義があります。

参照先 BCMFileIO::IdxBlock::dataClassID, BCMFileIO::IdxBlock::dataDir, BCMFileIO::LBHeader::dataType, BCMFileIO::IdxBlock::dataType, BCMFileIO::IdxBlock::extension, BCMFileIO::PartitionMapper::GetFDIDLists(), BCMFileIO::IdxBlock::isStepSubDir, BCMFileIO::LBHeader::kind, BCMFileIO::IdxBlock::kind, BCMFileIO::LB_FLOAT32, BCMFileIO::LB_FLOAT64, BCMFileIO::LB_INT16, BCMFileIO::LB_INT32, BCMFileIO::LB_INT64, BCMFileIO::LB_INT8, BCMFileIO::LB_UINT16, BCMFileIO::LB_UINT32, BCMFileIO::LB_UINT64, BCMFileIO::LB_UINT8, BCMFileIO::IdxBlock::prefix, BCMFileIO::IdxBlock::rootDir, BCMFileIO::LBHeader::size, BCMFileIO::LBHeader::vc, BCMFileIO::IdxBlock::vc, Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z (計 27 項目).

参照元 BCMFileIO::BCMFileLoader::LoadLeafBlock().

```
437         {
438                 using namespace std;
439                 vector<PartitionMapper::FDIDList> fdidlists;
440                 pmapper->GetFDIDLists(comm.Get_rank(), fdidlists);
441 
442                 Vec3i bsz = blockManager.getSize();
443 
444                 int did = 0;
445                 for(vector<PartitionMapper::FDIDList>::iterator file = fdidlists.begin(); file != fdidlists
    .end(); ++file){
446                         char filename[128];
447                         sprintf(filename, "%s_%010d_%06d.%s", ib->prefix.c_str(), step, file->FID, ib->
    extension.c_str());
448 
449                         string dirpath = ib->rootDir + ib->dataDir;
450                         if(ib->isStepSubDir){
451                                 char stepDirName[128];
452                                 sprintf(stepDirName, "%010d/", step);
453                                 dirpath += string(stepDirName);
```

```
454                         }
455
456                         string filepath = dirpath + string(filename);
457
458                         FILE *fp = NULL;
459                         if( (fp = fopen(filepath.c_str(), "rb")) == NULL ) {
460                                 Logger::Error("Cannnot open file (%s) [%s:%d]\n", filepath.
    c_str(), __FILE__, __LINE__);
461                                 return false;
462                         }
463
464                         bool isNeedSwap = false;
465                         LBHeader hdr;
466
467                         if( !LoadHeader(fp, hdr, isNeedSwap) ){
468                                 Logger::Error("%s is not leafBlock file [%s:%d]\n", filepath.
    c_str(), __FILE__, __LINE__);
469                                 return false;
470                         }
471
472                         if(hdr.kind != static_cast<unsigned char>(ib->kind) ){
473                                 Logger::Error("%s's kind(%d) is not corresponds IndexFile(%d)
     [%s:%d]\n", filepath.c_str(), hdr.kind, ib->kind, __FILE__, __LINE__);
474                                 return false;
475                         }
476
477                         if(hdr.dataType != static_cast<unsigned char>(ib->dataType)){
478                                 Logger::Error("%s's Type(%d) is not corresponds IndexFile(%d).
     [%s:%d]\n", filepath.c_str(), hdr.dataType, ib->dataType, __FILE__, __LINE__);
479                                 return false;
480                         }
481
482                         if(hdr.vc != ib->vc ){
483                                 Logger::Error("%s's vc(%d) is not corresponds IndexFile(%d).
     [%s:%d]\n", filepath.c_str(), hdr.vc, ib->vc, __FILE__, __LINE__);
484                                 return false;
485                         }
486
487                         if(hdr.size[0] != bsz.x || hdr.size[1] != bsz.y || hdr.size[2] != bsz.z){
488                                 Logger::Error("%s's size(%3d, %3d, %3d) is not corresponds
     IndexFile(%3d, %3d, %3d). [%s:%d]\n",
489                                         filepath.c_str(), hdr.size[0], hdr.size[1], hdr.size[2], bsz.x, bsz.y,
     bsz.z, __FILE__, __LINE__);
490                                 return false;
491                         }
492
493                         const static size_t typeByteTable[10] = { 1, 1, 2, 2, 4, 4, 8, 8, 4, 8 };
494                         size_t typeByte = typeByteTable[hdr.dataType];
495                         Vec3i fbsz( bsz.x + hdr.vc*2, bsz.y + hdr.vc*2, bsz.z + hdr.vc*2);
496
497                         fseek(fp, file->FDIDs[0] * typeByte * (fbsz.x * fbsz.y * fbsz.z) *
    static_cast<size_t>(ib->kind), SEEK_CUR);
498
499                         for(vector<int>::iterator fdid = file->FDIDs.begin(); fdid != file->FDIDs.end(); ++
    fdid){
500                                 for(int i = 0; i < static_cast<int>(ib->kind); i++){
501                                         int dcid = ib->dataClassID[i];
502                                         unsigned char* block =
    Load_BlockContents(fp, hdr, bsz, vc, isNeedSwap);
503
504                                         if     (ib->dataType == LB_INT8   ){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast< s8*>(block)); }
505                                         else if(ib->dataType == LB_UINT8  ){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast< u8*>(block)); }
506                                         else if(ib->dataType == LB_INT16  ){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast<s16*>(block)); }
507                                         else if(ib->dataType == LB_UINT16 ){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast<u16*>(block)); }
508                                         else if(ib->dataType == LB_INT32  ){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast<s32*>(block)); }
509                                         else if(ib->dataType == LB_UINT32 ){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast<u32*>(block)); }
510                                         else if(ib->dataType == LB_INT64  ){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast<s64*>(block)); }
511                                         else if(ib->dataType == LB_UINT64 ){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast<u64*>(block)); }
512                                         else if(ib->dataType == LB_FLOAT32){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast<f32*>(block)); }
513                                         else if(ib->dataType == LB_FLOAT64){
    CopyBufferToScalar3D(blockManager, dcid, did, vc, reinterpret_cast<f64*>(block)); }
514
515                                         delete [] block;
516                                 }
517
518                                 did++;
519                         }
520
```

```
521                         fclose(fp);
522                 }
523
524                 return true;
525         }
```

5.17.2.10 bool BCMFileIO::LeafBlockLoader::LoadHeader (FILE * *fp,* LBHeader & *hdr,* bool & *isNeedSwap*) [inline], [static], [private]

ヘッダを読み込む

LeafBlockLoader.cpp の 39 行目に定義があります。

参照先 BCMFileIO::LBHeader::bitWidth, BCMFileIO::BSwap16(), BCMFileIO::BSwap32(), BCMFileIO::BSwap64(), BCMFileIO::LBHeader::identifier, LEAFBLOCK_FILE_IDENTIFIER, BCMFileIO::LBHeader::numBlock, BCMFileIO::LBHeader::size, BCMFileIO::LBHeader::vc.

```
40         {
41                 fread(&hdr, sizeof(LBHeader), 1, fp);
42
43                 if( hdr.identifier != LEAFBLOCK_FILE_IDENTIFIER ){
44                         BSwap32(&hdr.identifier);
45
46                         if( hdr.identifier != LEAFBLOCK_FILE_IDENTIFIER ){
47                                 return false;
48                         }
49
50                         isNeedSwap = true;
51
52                         BSwap16(&hdr.bitWidth);
53                         BSwap32(&hdr.vc);
54                         BSwap32(&hdr.size[0]);
55                         BSwap32(&hdr.size[1]);
56                         BSwap32(&hdr.size[2]);
57                         BSwap64(&hdr.numBlock);
58                 }
59
60                 //Logger::Info("Header [bw : %d, vc : %d, sz : (%3d, %3d, %3d), nb : %d\n",
61                 //       hdr.bitWidth, hdr.vc, hdr.size[0], hdr.size[1], hdr.size[2], hdr.numBlock);
62
63                 return true;
64         }
```

このクラス詳解は次のファイルから抽出されました:

- LeafBlockLoader.h
- LeafBlockLoader.cpp

5.18 BCMFileIO::LeafBlockSaver クラス

LeafBlock ファイルを出力する関数群

`#include <LeafBlockSaver.h>`

静的公開メンバ関数

- static bool SaveCellID (const MPI::Intracomm &comm, const IdxBlock *ib, const Vec3i &size, const size_t numBlock, const unsigned char *datas, bool rle)
- static bool SaveData (const MPI::Intracomm &comm, const IdxBlock *ib, BlockManager &blockManager, const unsigned int step)

静的非公開メンバ関数

- template<typename T >
  static bool _SaveData (const MPI::Intracomm &comm, const IdxBlock *ib, BlockManager &blockManager, const unsigned int step)
- template<typename T >
  static bool CopyScalar3DToBuffer (BlockManager &blockManager, const int dataClassID, const int dataID, const int vc, T *buf)

5.18.1 詳解

LeafBlock ファイルを出力する関数群

LeafBlockSaver.h の 30 行目に定義があります。

5.18.2 関数詳解

5.18.2.1 template<typename T > bool BCMFileIO::LeafBlockSaver::_SaveData (const MPI::Intracomm & *comm*, const IdxBlock * *ib*, BlockManager & *blockManager*, const unsigned int *step*) [static], [private]

LeafBlockSaver.cpp の 212 行目に定義があります。

参照先 BCMFileIO::IdxBlock::bitWidth, BCMFileIO::IdxBlock::dataClassID, BCMFileIO::IdxBlock::dataDir, BCMFileIO::IdxBlock::dataType, BCMFileIO::IdxBlock::extension, BCMFileIO::LBHeader::identifier, BCMFileIO::IdxBlock::isStepSubDir, BCMFileIO::IdxBlock::kind, LEAFBLOCK_FILE_IDENTIFIER, BCMFileIO::IdxBlock::prefix, BCMFileIO::IdxBlock::rootDir, BCMFileIO::IdxBlock::vc.

```
216         {
217                 using namespace std;
218                 int rank = comm.Get_rank();
219
220                 Vec3i size = blockManager.getSize();
221
222                 LBHeader header;
223                 header.identifier = LEAFBLOCK_FILE_IDENTIFIER;
224                 header.kind       = static_cast<unsigned char>(ib->kind);
225                 header.dataType   = static_cast<unsigned char>(ib->dataType);
226                 header.bitWidth   = static_cast<unsigned short>(ib->bitWidth);
227                 header.vc         = ib->vc;
228                 header.size[0]    = size.x;
229                 header.size[1]    = size.y;
230                 header.size[2]    = size.z;
231                 header.numBlock   = blockManager.getNumBlock();
232
233                 int vc = ib->vc;
234                 const size_t sz = (size.x + vc*2) * (size.y + vc*2) * (size.z + vc*2);
235
236                 string outputDir = ib->rootDir + ib->dataDir;
237                 if(ib->isStepSubDir){
238                         char stepDirName[128];
239                         sprintf(stepDirName, "%010d/", step);
240                         outputDir += std::string(stepDirName);
241                 }
242
243                 FileSystemUtil::CreateDirectory(outputDir, outputDir.find("/
    ") == 0 ? true : false);
244
245                 FILE *fp = NULL;
246                 char filename[128];
247                 sprintf(filename, "%s_%010d_%06d.%s", ib->prefix.c_str(), step, rank, ib->extension.c_str()
    );
248
249                 string filepath = outputDir + string(filename);
250
251                 if( (fp = fopen(filepath.c_str(), "wb")) == NULL){
252                         Logger::Error("fileopen err <%s>. [%s:%d]\n", filepath.c_str(),
    __FILE__, __LINE__);
253                         return false;
```

```
254                 }
255
256                 fwrite(&header, sizeof(header), 1, fp);
257
258                 for(int id = 0; id < blockManager.getNumBlock(); ++id){
259                         for(int comp = 0; comp < static_cast<int>(ib->dataClassID.size()); comp++){
260                                 T* buf = new T[sz];
261                                 CopyScalar3DToBuffer(blockManager, ib->dataClassID[comp
    ], id, vc, buf);
262                                 fwrite(buf, sizeof(T), sz, fp);
263                                 delete buf;
264                         }
265                 }
266
267                 fclose(fp);
268
269                 return true;
270         }
```

5.18.2.2 template<typename T > bool BCMFileIO::LeafBlockSaver::CopyScalar3DToBuffer (BlockManager & *blockManager*, const int *dataClassID*, const int *dataID*, const int *vc*, T * *buf*) [static], [private]

LeafBlockSaver.cpp の 190 行目に定義があります。

参照先 Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z.

```
191         {
192                 Vec3i size = blockManager.getSize();
193
194                 BlockBase* block = blockManager.getBlock(dataID);
195                 Scalar3D<T>* mesh = dynamic_cast< Scalar3D<T>* >(block->getDataClass(dataClassID));
196                 T* data      = mesh->getData();
197                 Index3DS idx = mesh->getIndex();
198
199                 for(int z = -vc; z < size.z + vc; z++){
200                         for(int y = -vc; y < size.y + vc; y++){
201                                 for(int x = -vc; x < size.x + vc; x++){
202                                         size_t loc = ( (x+vc) + ((y+vc) + (z+vc) * (size.
    y + (vc*2))) * (size.x + (vc*2)) );
203                                         buf[loc] = data[idx(x, y, z)];
204                                 }
205                         }
206                 }
207                 return true;
208         }
```

5.18.2.3 bool BCMFileIO::LeafBlockSaver::SaveCellID (const MPI::Intracomm & *comm*, const IdxBlock * *ib*, const Vec3i & *size*, const size_t *numBlock*, const unsigned char * *datas*, bool *rle*) [static]

LeafBlock ファイル (CellID) の出力

引数

in	*comm*	MPI コミュニケータ
in	*ib*	ブロック情報
in	*size*	リーフブロックサイズ
in	*numBlock*	総ブロック数
in	*datas*	CellID が格納されたデータバッファ
in	*rle*	RLE 圧縮フラグ (true の場合RLE 圧縮を行う)

戻り値

成功した場合 true, 失敗した場合 false

LeafBlockSaver.cpp の 33 行目に定義があります。

参照先 BCMFileIO::IdxBlock::bitWidth, BCMFileIO::LBCellIDHeader::compSize, BCMFileIO::IdxBlock::dataDir, BCMFileIO::IdxBlock::dataType, BCMFileIO::IdxBlock::extension, BCMFileIO::LBHeader::identifier, BCMFileIO::IdxBlock::isGather, BCMFileIO::IdxBlock::kind, LEAFBLOCK_FILE_IDENTIFIER, BCMFileIO::LBCellIDHeader::numBlock, BCMFileIO::IdxBlock::prefix, BCMFileIO::IdxBlock::rootDir, BCMFileIO::IdxBlock::vc, Vec3class::Vec3< T >::x, Vec3class::Vec3< T >::y, Vec3class::Vec3< T >::z (計 16 項目).

参照元 BCMFileIO::BCMFileSaver::SaveLeafBlock().

```
39         {
40                 using namespace std;
41
42                 int rank = comm.Get_rank();
43
44                 // リーフブロックヘッダを準備
45                 LBHeader header;
46                 header.identifier = LEAFBLOCK_FILE_IDENTIFIER;
47                 header.kind       = static_cast<unsigned char>(ib->kind);
48                 header.dataType   = static_cast<unsigned char>(ib->dataType);
49                 header.bitWidth   = static_cast<unsigned short>(ib->bitWidth);
50                 header.vc         = ib->vc;
51                 header.size[0]    = size.x;
52                 header.size[1]    = size.y;
53                 header.size[2]    = size.z;
54                 header.numBlock   = 0; // temporary
55
56                 int vc = ib->vc;
57                 // 自プロセスの担当ブロックの保存用一時バッファサイズを計算
58                 const size_t tsz = (size.x + vc*2) * (size.y + vc*2) * (size.z + vc*2) * numBlock;
59
60                 // 自プロセスの担当ブロックを BitVoxel 化
61                 size_t bitVoxelSize = 0;
62                 bitVoxelCell* bitVoxel = BitVoxel::Compress(&bitVoxelSize,
    tsz, datas, ib->bitWidth);
63
64                 // リーフブロックの CellID ヘッダを準備
65                 LBCellIDHeader ch;
66                 ch.numBlock = numBlock;
67
68                 // BitVoxel バッファのサイズ (Byte 単位) を計算
69                 size_t bvs = bitVoxelSize * sizeof(bitVoxelCell);
70                 unsigned char* rleBuf = NULL;
71                 unsigned char* dp     = NULL;
72
73                 if( rle ){ // RLE 圧縮の場合
74                         size_t rleSize = 0;
75                         // BitVoxel を RLE 圧縮
76                         rleBuf = BCMRLE::Encode<bitVoxelCell, unsigned char>(bitVoxel, bvs, &rleSize);
77                         // リーフブロックの CellID ヘッダに圧縮サイズを記載
78                         ch.compSize = rleSize;
79                         dp = rleBuf;
80                 }else{ // RLE なしの場合
81                         // リーフブロックの CellID ヘッダに圧縮サイズ (=0) を記載
82                         ch.compSize = 0;
83                         dp = reinterpret_cast<unsigned char*>(bitVoxel);
84                 }
85
86                 if( ib->isGather ){ // GatherMode = "Gathered"
87                         int *numBlockTable      = NULL;
88                         int *leafBlockSizeTable = NULL;
89                         int bSz = ch.compSize == 0 ? bvs : static_cast<int>(ch.compSize);
90                         int nb  = static_cast<int>(ch.numBlock);
91                         if(rank == 0 ){
92                                 numBlockTable      = new int[comm.Get_size()]; // 各ランクのブロック数取得バッフ
ァ
93                                 leafBlockSizeTable = new int[comm.Get_size()]; // 各ランクのデータサイズ取得バッ
ファ
94                         }
95
96                         // 各ランクのブロック数を集約
97                         comm.Gather(&bSz, 1, MPI::INT, leafBlockSizeTable, 1, MPI::INT, 0);
98                         // 各ランクのデータサイズを集約
99                         comm.Gather(&nb,  1, MPI::INT, numBlockTable,      1, MPI::INT, 0);
100
101                         unsigned char* rcvBuf = NULL;
102                         int *displs = NULL;
103
104                         uint64_t allSz = 0;
105                         if( rank == 0 ){
106                                 displs = new int[comm.Get_size()];
107                                 for(int i = 0; i < comm.Get_size(); i++){
108                                         displs[i] = allSz;
109                                         allSz += leafBlockSizeTable[i];
110                                 }
111                                 rcvBuf = new unsigned char[allSz];
112                         }
```

```
113                         // 各ランクの圧縮済み CellID バッファを集約
114                         comm.Gatherv(dp, bSz, MPI::UNSIGNED_CHAR, rcvBuf, leafBlockSizeTable, displs,
    MPI::UNSIGNED_CHAR, 0);
115
116                         if( rank == 0 ){
117                                 char filename[128];
118                                 sprintf(filename, "%s.%s", ib->prefix.c_str(), ib->extension.c_str());
119                                 string filepath = ib->rootDir + ib->dataDir + string(filename);
120                                 FILE *fp = NULL;
121                                 if( (fp = fopen(filepath.c_str(), "wb")) == NULL) {
122                                         Logger::Error("fileopen error <%s>. [%s:%d]\n",
    filepath.c_str(), __FILE__, __LINE__);
123                                         return false;
124                                 }
125
126                                 for(int i = 0; i < comm.Get_size(); i++){
127                                         header.numBlock += numBlockTable[i];
128                                 }
129                                 // ヘッダを出力
130                                 fwrite(&header, sizeof(header), 1, fp);
131
132                                 uint64_t number_of_procs = comm.Get_size();
133                                 // プロセス数を出力
134                                 fwrite(&number_of_procs, sizeof(uint64_t), 1, fp);
135
136                                 for(int i = 0; i < comm.Get_size(); i++){
137                                         LBCellIDHeader ch;
138                                         ch.numBlock = numBlockTable[i];
139                                         ch.compSize = rle ? leafBlockSizeTable[i] : 0;
140                                         // 圧縮サイズを出力
141                                         fwrite(&ch, sizeof(LBCellIDHeader), 1, fp);
142                                 }
143                                 // 集約した CellID バッファを出力
144                                 fwrite(rcvBuf, sizeof(unsigned char), allSz, fp);
145
146                                 fclose(fp);
147
148                         }
149                         // 各メモリの解放
150                         if( rank == 0){
151                                 delete [] numBlockTable;
152                                 delete [] leafBlockSizeTable;
153                                 delete [] displs;
154                                 delete [] rcvBuf;
155                         }
156
157                 }else{ // GatherMode = "Distributed"
158
159                         // write file
160                         char filename[128];
161                         sprintf(filename, "%s_%06d.%s", ib->prefix.c_str(), rank, ib->extension.c_str());
162                         string filepath = ib->rootDir + ib->dataDir + string(filename);
163                         FILE *fp = NULL;
164                         if( (fp = fopen(filepath.c_str(), "wb")) == NULL) {
165                                 Logger::Error("fileopen error <%s>. [%s:%d]\n", filepath.c_str
    (), __FILE__, __LINE__);
166                                 return false;
167                         }
168
169                         header.numBlock = numBlock;
170
171                         fwrite(&header, sizeof(LBHeader),     1, fp);
172                         fwrite(&ch,     sizeof(LBCellIDHeader), 1, fp);
173
174                         size_t dsz = ch.compSize == 0 ? bvs : ch.compSize;
175
176                         fwrite(dp, sizeof(unsigned char), dsz, fp);
177
178                         fclose(fp);
179                 }
180
181                 delete [] bitVoxel;
182                 if( rleBuf != NULL ) delete [] rleBuf;
183
184                 return true;
185         }
```

5.18.2.4 bool BCMFileIO::LeafBlockSaver::SaveData (const MPI::Intracomm & *comm*, const IdxBlock * *ib*, BlockManager & *blockManager*, const unsigned int *step*) [static]

LeafBlock ファイル (物理量) の出力

引数

in	*comm*	MPI コミュニケータ
in	*ib*	ブロック情報
in	*blockManager*	ブロックマネージャ
in	*step*	出力タイムステップのインデックス番号

戻り値

成功した場合 true, 失敗した場合 false

LeafBlockSaver.cpp の 272 行目に定義があります。

参照先 BCMFileIO::IdxBlock::dataType, BCMFileIO::LB_FLOAT32, BCMFileIO::LB_FLOAT64, BCMFileIO::LB_INT16, BCMFileIO::LB_INT32, BCMFileIO::LB_INT64, BCMFileIO::LB_INT8, BCMFileIO::LB_UINT16, BCMFileIO::LB_UINT32, BCMFileIO::LB_UINT64, BCMFileIO::LB_UINT8.

参照元 BCMFileIO::BCMFileSaver::SaveLeafBlock().

```
276         {
277                 bool status = false;
278                 if     ( ib->dataType == LB_INT8   ) { status = _SaveData< s8>(comm, ib,
    blockManager, step); }
279                 else if( ib->dataType == LB_UINT8  ) { status = _SaveData< u8>(comm, ib,
    blockManager, step); }
280                 else if( ib->dataType == LB_INT16  ) { status = _SaveData<s16>(comm, ib,
    blockManager, step); }
281                 else if( ib->dataType == LB_UINT16 ) { status = _SaveData<u16>(comm, ib,
    blockManager, step); }
282                 else if( ib->dataType == LB_INT32  ) { status = _SaveData<s32>(comm, ib,
    blockManager, step); }
283                 else if( ib->dataType == LB_UINT32 ) { status = _SaveData<u32>(comm, ib,
    blockManager, step); }
284                 else if( ib->dataType == LB_INT64  ) { status = _SaveData<s64>(comm, ib,
    blockManager, step); }
285                 else if( ib->dataType == LB_UINT64 ) { status = _SaveData<u64>(comm, ib,
    blockManager, step); }
286                 else if( ib->dataType == LB_FLOAT32) { status = _SaveData<f32>(comm, ib,
    blockManager, step); }
287                 else if( ib->dataType == LB_FLOAT64) { status = _SaveData<f64>(comm, ib,
    blockManager, step); }
288                 else{
289                         Logger::Error("invalid DataType (%d)[%s:%d]\n", ib->dataType, __FILE__
    , __LINE__);
290                         return false;
291                 }
292
293                 return status;
294         }
```

このクラス詳解は次のファイルから抽出されました:

- LeafBlockSaver.h
- LeafBlockSaver.cpp

5.19 BCMFileIO::Logger クラス

ログ出力ユーティリティ

`#include <Logger.h>`

公開メンバ関数

- Logger ()

コンストラクタ

- ~Logger ()
  デストラクタ

静的公開メンバ関数

- static void Error (const char *format,...)
- static void Warn (const char *format,...)
- static void Info (const char *format,...)
- static void Debug (const char *format,...)

非公開型

- enum LOG_LEVEL { LOG_ERROR = 0, LOG_WARN, LOG_INFO, LOG_DEBUG }
  ログレベル

静的非公開メンバ関数

- static void Log (enum LOG_LEVEL level, const std::string &msg)

5.19.1 詳解

ログ出力ユーティリティ

Logger.h の 20 行目に定義があります。

5.19.2 列挙型メンバ詳解

5.19.2.1 enum BCMFileIO::Logger::LOG_LEVEL [private]

ログレベル

列挙値

LOG_ERROR

LOG_WARN

LOG_INFO

LOG_DEBUG

Logger.h の 24 行目に定義があります。

```
25                 {
26                         LOG_ERROR = 0,
27                         LOG_WARN,
28                         LOG_INFO,
29                         LOG_DEBUG
30                 };
```

5.19.3 構築子と解体子

5.19.3.1 BCMFileIO::Logger::Logger ( )

コンストラクタ

Logger.cpp の 28 行目に定義があります。

```
29          {
30          }
```

5.19.3.2 BCMFileIO::Logger::~Logger ( )

デストラクタ

Logger.cpp の 32 行目に定義があります。

```
33          {
34          }
```

5.19.4 関数詳解

5.19.4.1 void BCMFileIO::Logger::Debug (const char * *format*, ...) [static]

ログ出力 (デバッグ情報)

引数

in	*format*	ログ出力書式
in	...	ログ出力文字

Logger.cpp の 89 行目に定義があります。

参照先 Log(), LOG_DEBUG.

```
90          {
91                  char buf[256];
92                  va_list arg;
93                  va_start(arg, format);
94                  vsprintf(buf, format, arg);
95                  va_end(arg);
96
97                  Log(LOG_DEBUG, buf);
98          }
```

5.19.4.2 void BCMFileIO::Logger::Error (const char * *format*, ...) [static]

ログ出力 (エラー)

引数

in	*format*	ログ出力書式
in	...	ログ出力文字

Logger.cpp の 56 行目に定義があります。

参照先 Log(), LOG_ERROR.

参照元 BCMFileIO::BCMFileLoader::BCMFileLoader(), BCMFileIO::FileSystemUtil::CreateDirectory(), BCMFileIO::BCMFileLoader::CreateLeafBlock(), BCMFileIO::BCMFileLoader::LoadAdditionalIndex(), BCMFileIO::BCMFileLoader::LoadIndex(), BCMFileIO::BCMFileLoader::LoadIndexCellID(), BCMFileIO::BCMFileLoader::LoadIndexData(), BCMFileIO::BCMFileLoader::LoadIndexProc(), BCMFileIO::BCMFileLoader::LoadLeafBlock(), BCMFileIO::BCMFileLoader::LoadOctree(), BCMFileIO::BCMFileLoader::LoadOctreeFile(), BCMFileIO::BCMFileSaver::RegisterCellIDInformation(), BCMFileIO::BCMFileSaver::RegisterDataInformation(), BCMFileIO::BCMFileSaver::Save(), BCMFileIO::BCMFileSaver::SaveIndex(), BCMFileIO::BCMFileSaver::SaveIndexCellID(), BCMFileIO::BCMFileSaver::SaveIndexData(), BCMFileIO::BCMFileSaver::SaveIndexProc(), BCMFileIO::BCMFileSaver::SaveLeafBlock(), BCMFileIO::BCMFileSaver::SaveOctree() (計 20 項目).

```
57         {
58                 char buf[256];
59                 va_list arg;
60                 va_start(arg, format);
61                 vsprintf(buf, format, arg);
62                 va_end(arg);
63
64                 Log(LOG_ERROR, buf);
65         }
```

5.19.4.3 void BCMFileIO::Logger::Info (const char * *format*, ...) [static]

ログ出力 (情報)

引数

in	*format*	ログ出力書式
in	...	ログ出力文字

Logger.cpp の 78 行目に定義があります。

参照先 Log(), LOG_INFO.

```
79         {
80                 char buf[256];
81                 va_list arg;
82                 va_start(arg, format);
83                 vsprintf(buf, format, arg);
84                 va_end(arg);
85
86                 Log(LOG_INFO, buf);
87         }
```

5.19.4.4 void BCMFileIO::Logger::Log (enum LOG_LEVEL *level*, const std::string & *msg*) [static], [private]

ログ出力 (内部関数)

引数

in	*level*	ログレベル
in	*msg*	ログ出力メッセージ

Logger.cpp の 36 行目に定義があります。

参照元 Debug(), Error(), Info(), Warn().

```
37         {
38                 static const char *log_header[4] = {
39                         "[ERR]", "[WRN]", "[INF]", "[DBG]"
40                 };
41
42                 MPI::Intracomm& comm = MPI::COMM_WORLD;
43                 char rankstr[128];
```

```
44                 sprintf(rankstr, "[RANK:%6d]", comm.Get_rank());
45
46                 std::string logstr(rankstr);
47                 logstr += std::string(" ") + std::string(log_header[level]) + std::string(" ") + msg;
48
49         #ifdef _WIN32
50                 OutputDebugStringA(logstr.c_str());
51         #else
52                 printf("%s", logstr.c_str());
53         #endif
54         }
```

5.19.4.5 void BCMFileIO::Logger::Warn (const char * *format*, ...) [static]

ログ出力 (警告)

引数

in	*format*	ログ出力書式
in	...	ログ出力文字

Logger.cpp の 67 行目に定義があります。

参照先 Log(), LOG_WARN.

```
68         {
69                 char buf[256];
70                 va_list arg;
71                 va_start(arg, format);
72                 vsprintf(buf, format, arg);
73                 va_end(arg);
74
75                 Log(LOG_WARN, buf);
76         }
```

このクラス詳解は次のファイルから抽出されました:

- Logger.h
- Logger.cpp

5.20 BCMFileIO::OctHeader 構造体

Octree ファイルヘッダ構造体

`#include <BCMFileCommon.h>`

公開メンバ関数

- OctHeader ()

公開変数類

- unsigned int identifier
  エンディアン識別子
- double org [3]
  原点座標
- double rgn [3]

領域サイズ

- unsigned int rootDims [3]

  ルート分割数
- unsigned int maxLevel

  Octree **最大分割レベル**
- uint64_t numLeaf

  リーフノード数
- uint64_t padding

  16 **バイトアライメント用パディング**

5.20.1 詳解

Octree ファイルヘッダ構造体

BCMFileCommon.h の 49 行目に定義があります。

5.20.2 構築子と解体子

5.20.2.1 BCMFileIO::OctHeader::OctHeader ( ) [inline]

BCMFileCommon.h の 59 行目に定義があります。

```
59 : padding(0) {}
```

5.20.3 メンバ詳解

5.20.3.1 unsigned int BCMFileIO::OctHeader::identifier

エンディアン識別子

BCMFileCommon.h の 51 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadOctreeHeader(), BCMFileIO::BCMFileSaver::SaveOctree().

5.20.3.2 unsigned int BCMFileIO::OctHeader::maxLevel

Octree 最大分割レベル

BCMFileCommon.h の 55 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadOctreeHeader().

5.20.3.3 uint64_t BCMFileIO::OctHeader::numLeaf

リーフノード数

BCMFileCommon.h の 56 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadAdditionalIndex(), BCMFileIO::BCMFileLoader::LoadOctreeFile(), BCMFileIO::BCMFileLoader::LoadOctreeHeader().

5.20.3.4 double BCMFileIO::OctHeader::org[3]

原点座標

BCMFileCommon.h の 52 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadOctreeHeader().

5.20.3.5 uint64_t BCMFileIO::OctHeader::padding

16 バイトアライメント用パディング

BCMFileCommon.h の 57 行目に定義があります。

5.20.3.6 double BCMFileIO::OctHeader::rgn[3]

領域サイズ

BCMFileCommon.h の 53 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadOctreeHeader().

5.20.3.7 unsigned int BCMFileIO::OctHeader::rootDims[3]

ルート分割数

BCMFileCommon.h の 54 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadOctreeHeader().

この構造体詳解は次のファイルから抽出されました:

- BCMFileCommon.h

5.21 BCMFileIO::PartitionMapper クラス

MxN データロードのためのマッピングクラス

`#include <PartitionMapper.h>`

クラス

- struct FDIDList

  ファイル*ID* とファイル内のデータ*ID* リスト構造体

公開メンバ関数

- PartitionMapper (const int writeProcs, const int readProcs, const int numLeaf)
- ~PartitionMapper ()
  デストラクタ
- int GetReadProcs () const
  ファイル読込時の並列数を取得
- int GetWriteProcs () const
  ファイル出力時の並列数を取得
- int GetStart (const int rank) const
- int GetEnd (const int rank) const
- int GetFID (int did)
- int GetFDID (int did)
- bool GetFDIDLists (const int myRank, std::vector< FDIDList > &fdidlists)

非公開変数類

- Partition partW
  ファイル出力時の*Partition*
- Partition partR
  ファイル読込時の*Partition*
- const int writeProcs
  ファイル出力時の並列数
- const int readProcs
  ファイル読込時の並列数

5.21.1 詳解

MxN データロードのためのマッピングクラス

PartitionMapper.h の 26 行目に定義があります。

5.21.2 構築子と解体子

5.21.2.1 BCMFileIO::PartitionMapper::PartitionMapper (const int *writeProcs*, const int *readProcs*, const int *numLeaf*) [inline]

コンストラクタ

引数

in	*writeProcs*	ファイル出力時の並列数
in	*readProcs*	ファイル読込時の並列数
in	*numLeaf*	総リーフブロック数

PartitionMapper.h の 43 行目に定義があります。

```
44                          : partW(writeProcs, numLeaf),
45                            partR(readProcs, numLeaf),
46                            writeProcs(writeProcs),
47                            readProcs(readProcs)
48                  {
49
50                  }
```

5.21.2.2 BCMFileIO::PartitionMapper::~PartitionMapper ( ) [inline]

デストラクタ

PartitionMapper.h の 53 行目に定義があります。

```
54                  {
55
56                  }
```

5.21.3 関数詳解

5.21.3.1 int BCMFileIO::PartitionMapper::GetEnd (const int *rank*) const [inline]

ファイル読込時の末尾 did を取得

引数

in	*rank*	プロセス番号

戻り値

末尾 did

PartitionMapper.h の 76 行目に定義があります。

参照先 partR.

参照元 GetFDIDLists().

```
76 { return partR.getEnd(rank);   }
```

5.21.3.2 int BCMFileIO::PartitionMapper::GetFDID (int *did*) [inline]

グローバルデータID(did) のファイル相対データID(FDID) を取得

引数

in	*did*	did

戻り値

did のFDID

PartitionMapper.h の 90 行目に定義があります。

参照先 GetFID(), partW.

参照元 GetFDIDLists().

```
90 { return did - partW.getStart( this->GetFID(did) ); }
```

5.21.3.3 bool BCMFileIO::PartitionMapper::GetFDIDLists (const int *myRank*, std::vector< FDIDList > & *fdidlists*) [inline]

FDID リストのリストを取得

引数

in	*myRank*	プロセス番号
out	*fdidlists*	FDID リストのリスト

戻り値

成功した場合 true, 失敗した場合 false

FDIDList のリストを作成

PartitionMapper.h の 98 行目に定義があります。

参照先 GetEnd(), GetFDID(), GetFID(), GetStart().

参照元 BCMFileIO::LeafBlockLoader::LoadCellID(), BCMFileIO::LeafBlockLoader::LoadCellID_Gather(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadLeafBlock().

```
99                 {
100                         // 自ノードが担当するブロックのレンジを取得
101                         int didRange[2] = { GetStart(myRank), GetEnd(myRank) };
102
103                         // 自ノードが担当する DID が保存されている FID をリストアップ
104                         std::vector<int> fids;
105                         fids.reserve(didRange[1] - didRange[0]);
106                         for(int did = didRange[0]; did < didRange[1]; did++){
107                                 fids.push_back(GetFID(did));
108                         }
109
110                         // FID の重複削除
111                         sort(fids.begin(), fids.end());
112                         fids.erase( unique(fids.begin(), fids.end()), fids.end());
113
114
116                         fdidlists.resize(fids.size());
117
118                         for(size_t i = 0; i < fids.size(); i++){
119                                 fdidlists[i].FID = fids[i];
120                         }
121
122                         for(int did = didRange[0]; did < didRange[1]; did++){
123                                 fdidlists[ (GetFID(did) - fids[0]) ].FDIDs.push_back(
      GetFDID(did));
124                         }
125
126                         return true;
127                 }
```

5.21.3.4 int BCMFileIO::PartitionMapper::GetFID (int *did*) [inline]

グローバルデータID(did) が保存されているファイルのID(FID) を取得

引数

in	*did*	did

戻り値

did が保存されているFID

PartitionMapper.h の 83 行目に定義があります。

参照先 partW.

参照元 GetFDID(), GetFDIDLists().

```
83 { return partW.getRank(did); }
```

5.21.3.5 int BCMFileIO::PartitionMapper::GetReadProcs ( ) const [inline]

ファイル読込時の並列数を取得

PartitionMapper.h の 59 行目に定義があります。

参照先 readProcs.

```
59 { return readProcs; }
```

5.21.3.6 int BCMFileIO::PartitionMapper::GetStart (const int *rank*) const [inline]

ファイル読込時の先頭 did を取得

引数

in	*rank*	プロセス番号

戻り値

先頭 did

PartitionMapper.h の 69 行目に定義があります。

参照先 partR.

参照元 GetFDIDLists().

```
69 { return partR.getStart(rank); }
```

5.21.3.7 int BCMFileIO::PartitionMapper::GetWriteProcs ( ) const [inline]

ファイル出力時の並列数を取得

PartitionMapper.h の 62 行目に定義があります。

参照先 writeProcs.

参照元 BCMFileIO::LeafBlockLoader::LoadCellID_Gather().

```
62 { return writeProcs; }
```

5.21.4 メンバ詳解

5.21.4.1 Partition BCMFileIO::PartitionMapper::partR [private]

ファイル読込時のPartition

PartitionMapper.h の 132 行目に定義があります。

参照元 GetEnd(), GetStart().

5.21.4.2 Partition BCMFileIO::PartitionMapper::partW [private]

ファイル出力時のPartition

PartitionMapper.h の 131 行目に定義があります。

参照元 GetFDID(), GetFID().

5.21.4.3 const int BCMFileIO::PartitionMapper::readProcs [private]

ファイル読込時の並列数

PartitionMapper.h の 134 行目に定義があります。

参照元 GetReadProcs().

5.21.4.4 const int BCMFileIO::PartitionMapper::writeProcs [private]

ファイル出力時の並列数

PartitionMapper.h の 133 行目に定義があります。

参照元 GetWriteProcs().

このクラス詳解は次のファイルから抽出されました:

- PartitionMapper.h

5.22 Vec3class::Vec3< T > クラステンプレート

`#include <Vec3.h>`

公開メンバ関数

- Vec3 (T v=0)
- Vec3 (T _x, T _y, T _z)
- Vec3 (const T v[3])
- Vec3< T > & assign (T _x, T _y, T _z)
- operator T * ()
- operator const T * () const
- T * ptr ()
- const T * ptr () const
- T & operator[] (const AxisEnum &axis)
- const T & operator[] (const AxisEnum &axis) const
- Vec3< T > & operator+= (const Vec3< T > &v)
- Vec3< T > & operator-= (const Vec3< T > &v)
- Vec3< T > & operator*= (const Vec3< T > &v)
- Vec3< T > & operator/= (const Vec3< T > &v)
- Vec3< T > & operator*= (T s)
- Vec3< T > & operator/= (T s)
- Vec3< T > operator+ (const Vec3< T > &v) const

- Vec3< T > operator- (const Vec3< T > &v) const
- Vec3< T > operator* (const Vec3< T > &v) const
- Vec3< T > operator/ (const Vec3< T > &v) const
- Vec3< T > operator* (T s) const
- Vec3< T > operator/ (T s) const
- Vec3< T > operator- () const
- bool operator== (const Vec3< T > &v) const
- bool operator!= (const Vec3< T > &v) const
- T lengthSquared () const
- T length () const
- Vec3< T > & normalize ()
- Vec3< T > & normalize (T *len)
- T average () const

静的公開メンバ関数

- static Vec3< T > xaxis ()
- static Vec3< T > yaxis ()
- static Vec3< T > zaxis ()

公開変数類

- T t [3]
- T x
- T y
- T z

5.22.1 詳解

template<typename T>class Vec3class::Vec3< T >

Vec3.h の 51 行目に定義があります。

5.22.2 構築子と解体子

5.22.2.1 template<typename T> Vec3class::Vec3< T >::Vec3 (T *v* = 0) [inline]

Vec3.h の 53 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
53 { t[0] = t[1] = t[2] = v; }
```

5.22.2.2 template<typename T> Vec3class::Vec3< T >::Vec3 (T *_x*, T *_y*, T *_z*) [inline]

Vec3.h の 54 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
54 { t[0]=_x; t[1]=_y; t[2]=_z; }
```

5.22.2.3 template<typename T> Vec3class::Vec3< T >::Vec3 (const T *v[3]*) [inline]

Vec3.h の 55 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
55 { t[0] = v[0]; t[1] = v[1]; t[2] = v[2]; }
```

5.22.3 関数詳解

5.22.3.1 template<typename T> Vec3<T>& Vec3class::Vec3< T >::assign (T *_x*, T *_y*, T *_z*) [inline]

Vec3.h の 57 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
57                                          {
58                 t[0]=_x; t[1]=_y; t[2]=_z;
59                 return *this;
60         }
```

5.22.3.2 template<typename T> T Vec3class::Vec3< T >::average ( ) const [inline]

Vec3.h の 173 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
173 { return (t[0] + t[1] + t[2])/3.f; }
```

5.22.3.3 template<typename T> T Vec3class::Vec3< T >::length ( ) const [inline]

Vec3.h の 155 行目に定義があります。

参照先 Vec3class::Vec3< T >::lengthSquared().

参照元 Vec3class::Vec3< T >::normalize().

```
155 { return sqrt(lengthSquared()); }
```

5.22.3.4 template<typename T> T Vec3class::Vec3< T >::lengthSquared ( ) const [inline]

Vec3.h の 151 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

参照元 Vec3class::Vec3< T >::length(), Vec3class::lessVec3f().

```
151                                 {
152                 return t[0] * t[0] + t[1] * t[1] + t[2] *t [2];
153         }
```

5.22.3.5 template<typename T> Vec3<T>& Vec3class::Vec3< T >::normalize ( ) [inline]

Vec3.h の 157 行目に定義があります。

参照先 Vec3class::Vec3< T >::length().

```
157                             {
158                 T len = length();
159                 if (len != 0)
160                         return *this /= len;
161                 else
162                         return *this;
163         }
```

5.22.3.6 template<typename T> Vec3<T>& Vec3class::Vec3< T >::normalize (T * *len*) [inline]

Vec3.h の 165 行目に定義があります。

参照先 Vec3class::Vec3< T >::length().

```
165                                     {
166                 *len = length();
167                 if (*len != 0)
168                         return *this /= *len;
169                 else
170                         return *this;
171         }
```

5.22.3.7 template<typename T> Vec3class::Vec3< T >::operator const T * ( ) const [inline]

Vec3.h の 63 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
63 { return &t[0]; }
```

5.22.3.8 template<typename T> Vec3class::Vec3< T >::operator T * ( ) [inline]

Vec3.h の 62 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
62 { return &t[0]; }
```

5.22.3.9 template<typename T> bool Vec3class::Vec3< T >::operator!= (const Vec3< T > & *v*) const [inline]

Vec3.h の 143 行目に定義があります。

```
143                                                     {
144                 return !(*this == v);
145         }
```

5.22.3.10 template<typename T> Vec3<T> Vec3class::Vec3< T >::operator*(const Vec3< T > & *v*) const [inline]

Vec3.h の 118 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
118                                                 {
119                 return Vec3<T>(t[0] * v.t[0], t[1] * v.t[1], t[2] * v.t[2]);
120         }
```

5.22.3.11 template<typename T> Vec3<T> Vec3class::Vec3< T >::operator*(T *s*) const [inline]

Vec3.h の 126 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
126                                     {
127                 return Vec3<T>(t[0] * s, t[1] * s, t[2] * s);
128         }
```

5.22.3.12 template<typename T> Vec3<T>& Vec3class::Vec3< T >::operator*=(const Vec3< T > & *v*) [inline]

Vec3.h の 89 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
89                                                  {
90                 t[0] *= v.t[0]; t[1] *= v.t[1]; t[2] *= v.t[2];
91                 return *this;
92         }
```

5.22.3.13 template<typename T> Vec3<T>& Vec3class::Vec3< T >::operator*=(T *s*) [inline]

Vec3.h の 99 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
99                                      {
100                 t[0] *= s; t[1] *= s; t[2] *= s;
101                 return *this;
102         }
```

5.22.3.14 template<typename T> Vec3<T> Vec3class::Vec3< T >::operator+(const Vec3< T > & *v*) const [inline]

Vec3.h の 110 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
110                                                 {
111                 return Vec3<T>(t[0] + v.t[0], t[1] + v.t[1], t[2] + v.t[2]);
112         }
```

5.22.3.15 template<typename T> Vec3<T>& Vec3class::Vec3< T >::operator+= (const Vec3< T > & *v*) [inline]

Vec3.h の 79 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
79                                                {
80                 t[0] += v.t[0]; t[1] += v.t[1]; t[2] += v.t[2];
81                 return *this;
82         }
```

5.22.3.16 template<typename T> Vec3<T> Vec3class::Vec3< T >::operator- (const Vec3< T > & *v*) const [inline]

Vec3.h の 114 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
114                                                  {
115                 return Vec3<T>(t[0] - v.t[0], t[1] - v.t[1], t[2] - v.t[2]);
116         }
```

5.22.3.17 template<typename T> Vec3<T> Vec3class::Vec3< T >::operator- ( ) const [inline]

Vec3.h の 135 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
135                                  {
136                 return Vec3<T>(-t[0], -t[1], -t[2]);
137         }
```

5.22.3.18 template<typename T> Vec3<T>& Vec3class::Vec3< T >::operator-= (const Vec3< T > & *v*) [inline]

Vec3.h の 84 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
84                                                {
85                 t[0] -= v.t[0]; t[1] -= v.t[1]; t[2] -= v.t[2];
86                 return *this;
87         }
```

5.22.3.19 template<typename T> Vec3<T> Vec3class::Vec3< T >::operator/ (const Vec3< T > & *v*) const [inline]

Vec3.h の 122 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
122                                                  {
123                 return Vec3<T>(t[0] / v.t[0], t[1] / v.t[1], t[2] / v.t[2]);
124         }
```

5.22.3.20 template<typename T> Vec3<T> Vec3class::Vec3< T >::operator/ (T *s*) const [inline]

Vec3.h の 130 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
130                                     {
131                 T inv = 1./s;
132                 return Vec3<T>(t[0] * inv, t[1] * inv, t[2] * inv);
133         }
```

5.22.3.21 template<typename T> Vec3<T>& Vec3class::Vec3< T >::operator/= (const Vec3< T > & *v*) [inline]

Vec3.h の 94 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
94                                              {
95                 t[0] /= v.t[0]; t[1]/= v.t[1]; t[2] /= v.t[2];
96                 return *this;
97         }
```

5.22.3.22 template<typename T> Vec3<T>& Vec3class::Vec3< T >::operator/= (T *s*) [inline]

Vec3.h の 104 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
104                                     {
105                 T inv = 1./s;
106                 t[0] *= inv; t[1] *= inv; t[2] *= inv;
107                 return *this;
108         }
```

5.22.3.23 template<typename T> bool Vec3class::Vec3< T >::operator== (const Vec3< T > & *v*) const [inline]

Vec3.h の 139 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
139                                                     {
140                 return t[0] == v.t[0] && t[1] == v.t[1] && t[2] == v.t[2];
141         }
```

5.22.3.24 template<typename T> T& Vec3class::Vec3< T >::operator[] (const AxisEnum & *axis*) [inline]

Vec3.h の 69 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
69                                              {
70                 return t[axis];
71         }
```

5.22.3.25 template<typename T> const T& Vec3class::Vec3< T >::operator[] (const AxisEnum & *axis*) const [inline]

Vec3.h の 72 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
72                                                      {
73                 return t[axis];
74         }
```

5.22.3.26 template<typename T> T* Vec3class::Vec3< T >::ptr ( ) [inline]

Vec3.h の 64 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
64 { return &t[0]; }
```

5.22.3.27 template<typename T> const T* Vec3class::Vec3< T >::ptr ( ) const [inline]

Vec3.h の 65 行目に定義があります。

参照先 Vec3class::Vec3< T >::t.

```
65 { return &t[0]; }
```

5.22.3.28 template<typename T> static Vec3<T> Vec3class::Vec3< T >::xaxis ( ) [inline], [static]

Vec3.h の 147 行目に定義があります。

```
147 { return Vec3<T>(1, 0, 0); }
```

5.22.3.29 template<typename T> static Vec3<T> Vec3class::Vec3< T >::yaxis ( ) [inline], [static]

Vec3.h の 148 行目に定義があります。

```
148 { return Vec3<T>(0, 1, 0); }
```

5.22.3.30 template<typename T> static Vec3<T> Vec3class::Vec3< T >::zaxis ( ) [inline], [static]

Vec3.h の 149 行目に定義があります。

```
149 { return Vec3<T>(0, 0, 1); }
```

5.22.4 メンバ詳解

5.22.4.1 template<typename T> T Vec3class::Vec3< T >::t[3]

Vec3.h の 175 行目に定義があります。

参照元 Vec3class::Vec3< T >::assign(), Vec3class::Vec3< T >::average(), Vec3class::cross(), Vec3class::dot(), Vec3class::Vec3< T >::lengthSquared(), Vec3class::Vec3< T >::operator const T *(), Vec3class::Vec3< T >::operator T *(), Vec3class::Vec3< T >::operator*(), Vec3class::operator*(), Vec3class::Vec3< T >::operator*=(), Vec3class::Vec3< T >::operator+(), Vec3class::Vec3< T >::operator+=(), Vec3class::Vec3< T >::operator-(), Vec3class::Vec3< T >::operator-=(), Vec3class::Vec3< T >::operator/(), Vec3class::Vec3< T >::operator/=(), Vec3class::Vec3< T >::operator==(), Vec3class::Vec3< T >::operator[](), Vec3class::Vec3< T >::ptr(), Vec3class::Vec3< T >::Vec3() (計 20 項目).

5.22.4.2 template<typename T> T Vec3class::Vec3< T >::x

Vec3.h の 176 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::CopyBufferToScalar3D(), BCMFileIO::LeafBlockSaver::CopyScalar3DToBuffer(), BCMFileIO::BCMFileSaver::GetCellIDBlock(), BCMFileIO::LeafBlockLoader::Load_BlockContents(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadLeafBlock(), BCMFileIO::BCMFileLoader::LoadOctree(), BCMFileIO::BCMFileLoader::ReadVec3(), BCMFileIO::LeafBlockSaver::SaveCellID(), BCMFileIO::BCMFileSaver::SaveOctree().

5.22.4.3 template<typename T> T Vec3class::Vec3< T >::y

Vec3.h の 176 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::CopyBufferToScalar3D(), BCMFileIO::LeafBlockSaver::CopyScalar3DToBuffer(), BCMFileIO::BCMFileSaver::GetCellIDBlock(), BCMFileIO::LeafBlockLoader::Load_BlockContents(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadLeafBlock(), BCMFileIO::BCMFileLoader::LoadOctree(), BCMFileIO::BCMFileLoader::ReadVec3(), BCMFileIO::LeafBlockSaver::SaveCellID(), BCMFileIO::BCMFileSaver::SaveOctree().

5.22.4.4 template<typename T> T Vec3class::Vec3< T >::z

Vec3.h の 176 行目に定義があります。

参照元 BCMFileIO::LeafBlockLoader::CopyBufferToScalar3D(), BCMFileIO::LeafBlockSaver::CopyScalar3DToBuffer(), BCMFileIO::BCMFileSaver::GetCellIDBlock(), BCMFileIO::LeafBlockLoader::Load_BlockContents(), BCMFileIO::LeafBlockLoader::LoadData(), BCMFileIO::BCMFileLoader::LoadLeafBlock(), BCMFileIO::BCMFileLoader::LoadOctree(), BCMFileIO::BCMFileLoader::ReadVec3(), BCMFileIO::LeafBlockSaver::SaveCellID(), BCMFileIO::BCMFileSaver::SaveOctree().

このクラス詳解は次のファイルから抽出されました:

- Vec3.h

Chapter 6

ファイル詳解

6.1 BCMFileCommon.h ファイル

BCM ファイルIO 用共通クラス群

```
#include <limits.h>
#include <vector>
#include <string>
#include <list>
#include "BitVoxel.h"
#include <stdint.h>
```

BCMFileCommon.h の依存先関係図: 被依存関係図:

クラス

- struct BCMFileIO::OctHeader
  Octree **ファイルヘッダ構造体**
- struct BCMFileIO::LBHeader
  LeafBlock **ファイルヘッダ構造体**
- struct BCMFileIO::LBCellIDHeader
  LeafBlock **の***CellID* **ヘッダ構造体**
- struct BCMFileIO::GridRleCode
  RLE **圧縮符号の走査用構造体**
- struct BCMFileIO::IdxUnit
  インデックスファイル用単位系情報
- struct BCMFileIO::IdxProc
  インデックスファイル用プロセス情報

名前空間

- BCMFileIO

マクロ定義

- #define OCTREE_FILE_IDENTIFIER (('O' | ('C' << 8) | ('0' << 16) | ('1' << 24)))
  Octree **ファイルのエンディアン識別子** *(OC01)*

- #define LEAFBLOCK_FILE_IDENTIFIER (('L' | ('B' << 8) | ('0' << 16) | ('1' << 24)))

  LeafBlock **ファイルのエンディアン識別子** *(LB01)*
- #define ALIGNMENT

列挙型

- enum BCMFileIO::LB_KIND {
  BCMFileIO::LB_CELLID = 0, BCMFileIO::LB_SCALAR = 1, BCMFileIO::LB_VECTOR3 = 3, BCMFileIO::LB_VECTOR4 = 4,
  BCMFileIO::LB_VECTOR6 = 6, BCMFileIO::LB_TENSOR = 9 }

  リーフブロックデータタイプ
- enum BCMFileIO::LB_DATA_TYPE {
  BCMFileIO::LB_INT8 = 0, BCMFileIO::LB_UINT8 = 1, BCMFileIO::LB_INT16 = 2, BCMFileIO::LB_UINT16 = 3,
  BCMFileIO::LB_INT32 = 4, BCMFileIO::LB_UINT32 = 5, BCMFileIO::LB_INT64 = 6, BCMFileIO::LB_UINT64 = 7,
  BCMFileIO::LB_FLOAT32 = 8, BCMFileIO::LB_FLOAT64 = 9 }

  リーフセルのデータ識別子

関数

- static void BCMFileIO::BSwap16 (void *a)

  2byte **用エンディアンスワップ**
- static void BCMFileIO::BSwap32 (void *a)

  4byte **用エンディアンスワップ**
- static void BCMFileIO::BSwap64 (void *a)

  8byte **用エンディアンスワップ**

変数

- struct BCMFileIO::OctHeader BCMFileIO::ALIGNMENT

6.1.1 詳解

BCM ファイルIO 用共通クラス群

BCMFileCommon.h に定義があります。

6.1.2 マクロ定義詳解

6.1.2.1 #define ALIGNMENT

BCMFileCommon.h の 45 行目に定義があります。

6.1.2.2 #define LEAFBLOCK_FILE_IDENTIFIER (('L' | ('B' << 8) | ('0' << 16) | ('1' << 24)))

LeafBlock ファイルのエンディアン識別子 (LB01)

BCMFileCommon.h の 36 行目に定義があります。

参照元 BCMFileIO::LeafBlockSaver::_SaveData(), BCMFileIO::LeafBlockLoader::LoadHeader(), BCMFileIO::LeafBlockSaver::SaveCellID().

6.1.2.3 #define OCTREE_FILE_IDENTIFIER (('O' | ('C' << 8) | ('0' << 16) | ('1' << 24)))

Octree ファイルのエンディアン識別子 (OC01)

BCMFileCommon.h の 33 行目に定義があります。

参照元 BCMFileIO::BCMFileLoader::LoadOctreeHeader(), BCMFileIO::BCMFileSaver::SaveOctree().

6.2 BCMFileLoader.cpp ファイル

BCM ファイルを読み込むクラス

```
#include "BCMFileLoader.h"
#include "BCMOctree.h"
#include "RootGrid.h"
#include "BlockManager.h"
#include "BoundaryConditionSetterBase.h"
#include "Block.h"
#include "BlockFactory.h"
#include "PartitionMapper.h"
#include "Scalar3D.h"
#include "Vector3D.h"
#include "Scalar3DUpdater.h"
#include "Vector3DUpdater.h"
#include "Vec3.h"
#include "TextParser.h"
#include <vector>
#include <string>
#include "BCMFileCommon.h"
#include "LeafBlockLoader.h"
#include "FileSystemUtil.h"
#include "ErrorUtil.h"
#include "Logger.h"
#include "BCMTypes.h"
```

BCMFileLoader.cpp の依存先関係図:

名前空間

- BCMFileIO

マクロ定義

- #define OCTREE_LOAD_ONLY_MASTER

型定義

- typedef
  LeafBlockLoader::CellIDCapsule BCMFileIO::CellIDCapsule

6.2.1 詳解

BCM ファイルを読み込むクラス

BCMFileLoader.cpp に定義があります。

6.2.2 マクロ定義詳解

6.2.2.1 #define OCTREE_LOAD_ONLY_MASTER

BCMFileLoader.cpp の 43 行目に定義があります。

6.3 BCMFileLoader.h ファイル

BCM ファイルを読み込むクラス

```
#include <mpi.h>
#include <string>
#include <vector>
#include "Vec3.h"
#include "BCMFileCommon.h"
#include "IdxBlock.h"
#include "IdxStep.h"
#include "Pedigree.h"
```

BCMFileLoader.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::BCMFileLoader

  BCM **ファイルを読み込むクラス**

名前空間

- BCMFileIO

6.3.1 詳解

BCM ファイルを読み込むクラス

BCMFileLoader.h に定義があります。

6.4 BCMFileSaver.cpp ファイル

BCM ファイルを出力するクラス

```
#include "BCMOctree.h"
#include "RootGrid.h"
#include "BlockManager.h"
#include "Partition.h"
#include <cstring>
#include <vector>
#include <string>
#include <fstream>
#include <sstream>
#include "ErrorUtil.h"
#include "FileSystemUtil.h"
#include "BCMFileCommon.h"
#include "BCMFileSaver.h"
#include "LeafBlockSaver.h"
#include "Logger.h"
#include "Scalar3D.h"
#include "Vector3D.h"
#include "Vec3.h"
```

BCMFileSaver.cpp の依存先関係図:

名前空間

- BCMFileIO

マクロ定義

- #define ENABLE_RLE_ENCODE

6.4.1 詳解

BCM ファイルを出力するクラス

BCMFileSaver.cpp に定義があります。

6.4.2 マクロ定義詳解

6.4.2.1 #define ENABLE_RLE_ENCODE

BCMFileSaver.cpp の 37 行目に定義があります。

6.5 BCMFileSaver.h ファイル

BCM ファイルを出力するクラス

```
#include <mpi.h>
#include <string>
#include <vector>
#include "Vec3.h"
#include "BCMFileCommon.h"
#include "IdxBlock.h"
#include "IdxStep.h"
```

BCMFileSaver.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::BCMFileSaver
  BCM **ファイルを出力するクラス**

名前空間

- BCMFileIO

6.5.1 詳解

BCM ファイルを出力するクラス

BCMFileSaver.h に定義があります。

6.6 BCMRLE.h ファイル

ランレングスによる圧縮/展開ライブラリ

被依存関係図:

クラス

- class BCMFileIO::BCMRLE
  ランレングスによる圧縮/展開ライブラリ

名前空間

- BCMFileIO

マクロ定義

- #define ALIGNMENT
- #define ALIGNMENT

6.6.1 詳解

ランレングスによる圧縮/展開ライブラリ

BCMRLE.h に定義があります。

6.6.2 マクロ定義詳解

6.6.2.1 #define ALIGNMENT

6.6.2.2 #define ALIGNMENT

6.7 BCMTypes.h ファイル

クロスプラットホームのデータ型宣言

```
#include <stdint.h>
```

BCMTypes.h の依存先関係図: 被依存関係図:

型定義

- typedef bool b8
  論理型
- typedef char s8
  符号付き *8bit* **整数型**
- typedef unsigned char u8
  符号なし *8bit* **整数型**
- typedef int16_t s16
  符号付き *16bit* **整数型**
- typedef uint16_t u16
  符号なし *16bit* **整数型**
- typedef int32_t s32
  符号付き *32bit* **整数型**
- typedef uint32_t u32
  符号なし *32bit* **整数型**
- typedef int64_t s64
  符号付き *64bit* **整数型**
- typedef uint64_t u64
  符号なし *64bit* **整数型**
- typedef float f32
  32bit **浮動小数点** *(***単精度浮動小数点***)*
- typedef double f64
  64bit **浮動小数点** *(***倍精度浮動小数点***)*

6.7.1 詳解

クロスプラットホームのデータ型宣言

BCMTypes.h に定義があります。

6.7.2 型定義詳解

6.7.2.1 typedef bool b8

論理型
BCMTypes.h の 34 行目に定義があります。

6.7.2.2 typedef float f32

32bit 浮動小数点 (単精度浮動小数点)
BCMTypes.h の 43 行目に定義があります。

6.7.2.3 typedef double f64

64bit 浮動小数点 (倍精度浮動小数点)
BCMTypes.h の 44 行目に定義があります。

6.7.2.4 typedef int16_t s16

符号付き 16bit 整数型
BCMTypes.h の 37 行目に定義があります。

6.7.2.5 typedef int32_t s32

符号付き 32bit 整数型
BCMTypes.h の 39 行目に定義があります。

6.7.2.6 typedef int64_t s64

符号付き 64bit 整数型
BCMTypes.h の 41 行目に定義があります。

6.7.2.7 typedef char s8

符号付き 8bit 整数型
BCMTypes.h の 35 行目に定義があります。

6.7.2.8 typedef uint16_t u16

符号なし 16bit 整数型

BCMTypes.h の 38 行目に定義があります。

6.7.2.9 typedef uint32_t u32

符号なし 32bit 整数型

BCMTypes.h の 40 行目に定義があります。

6.7.2.10 typedef uint64_t u64

符号なし 64bit 整数型

BCMTypes.h の 42 行目に定義があります。

6.7.2.11 typedef unsigned char u8

符号なし 8bit 整数型

BCMTypes.h の 36 行目に定義があります。

6.8 BitVoxel.cpp ファイル

ビットボクセル圧縮/展開ライブラリ

```
#include <cstring>
#include "BitVoxel.h"
```

BitVoxel.cpp の依存先関係図:

名前空間

- BCMFileIO

型定義

- typedef unsigned int BCMFileIO::bitVoxelCell

6.8.1 詳解

ビットボクセル圧縮/展開ライブラリ

BitVoxel.cpp に定義があります。

6.9 BitVoxel.h ファイル

ビットボクセル圧縮/展開ライブラリ

```
#include <cstdlib>
```

BitVoxel.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::BitVoxel
  ビットボクセル圧縮/展開ライブラリ

名前空間

- BCMFileIO

6.9.1 詳解

ビットボクセル圧縮/展開ライブラリ

BitVoxel.h に定義があります。

6.10 DirUtil.cpp ファイル

```
#include "FileSystemUtil.h"
#include <vector>
#include <string>
#include <algorithm>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dirent.h>
#include "DirUtil.h"
#include "ErrorUtil.h"
```

DirUtil.cpp の依存先関係図:

名前空間

- BCMFileIO

6.11 DirUtil.h ファイル

ディレクトリ操作ユーティリティ

```
#include <vector>
#include <string>
```

DirUtil.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::DirUtil

  ディレクトリ操作ユーティリティ

名前空間

- BCMFileIO

6.11.1 詳解

ディレクトリ操作ユーティリティ

DirUtil.h に定義があります。

6.12 ErrorUtil.cpp ファイル

エラー処理関連のユーティリティ

```
#include "ErrorUtil.h"
#include <stdio.h>
#include <stdarg.h>
#include <string>
```

ErrorUtil.cpp の依存先関係図:

名前空間

- BCMFileIO

6.12.1 詳解

エラー処理関連のユーティリティ

ErrorUtil.cpp に定義があります。

6.13 ErrorUtil.h ファイル

エラー処理関連のユーティリティ

```
#include <mpi.h>
```

ErrorUtil.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::ErrorUtil

  エラー処理関連のユーティリティ

名前空間

- BCMFileIO

6.13.1 詳解

エラー処理関連のユーティリティ

ErrorUtil.h に定義があります。

6.14 FileSystemUtil.h ファイル

ファイル操作関連ユーティリティ

```
#include <algorithm>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dirent.h>
#include <string>
#include <vector>
#include "DirUtil.h"
#include "Logger.h"
```

FileSystemUtil.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::FileSystemUtil

  ファイル操作関連ユーティリティ

名前空間

- BCMFileIO

6.14.1 詳解

ファイル操作関連ユーティリティ

FileSystemUtil.h に定義があります。

6.15 hdmVersion.h ファイル

マクロ定義

- #define HDM_VERSION_NO "0.3.0"
- #define HDM_REVISION "20140317_1650"

6.15.1 詳解

HDM バージョン情報のヘッダーファイル

hdmVersion.h に定義があります。

6.15.2 マクロ定義詳解

6.15.2.1 #define HDM_REVISION "20140317_1650"

HDM ライブラリのリビジョン

hdmVersion.h の 21 行目に定義があります。

6.15.2.2 #define HDM_VERSION_NO "0.3.0"

HDM ライブラリのバージョン

hdmVersion.h の 18 行目に定義があります。

6.16 IdxBlock.h ファイル

```
#include <vector>
#include <string>
#include "BCMFileCommon.h"
#include "IdxStep.h"
```

IdxBlock.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::IdxBlock
  インデックスファイル用ブロック情報クラス

名前空間

- BCMFileIO

6.17 IdxStep.cpp ファイル

インデックスファイル用タイムステップ情報クラス

```
#include "IdxStep.h"
```

IdxStep.cpp の依存先関係図:

名前空間

- BCMFileIO

6.17.1 詳解

インデックスファイル用タイムステップ情報クラス

IdxStep.cpp に定義があります。

6.18 IdxStep.h ファイル

インデックスファイル用タイムステップ情報クラス

```
#include <vector>
#include <list>
#include "BitVoxel.h"
```

IdxStep.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::IdxStep

  インデックスファイル用タイムステップ情報

名前空間

- BCMFileIO

6.18.1 詳解

インデックスファイル用タイムステップ情報クラス

IdxStep.h に定義があります。

6.19 LeafBlockLoader.cpp ファイル

LeafBlock ファイルを読み込むクラス

```
#include <vector>
#include <string>
#include "LeafBlockLoader.h"
#include "BitVoxel.h"
#include "BCMRLE.h"
#include "ErrorUtil.h"
#include "Logger.h"
#include "BCMTypes.h"
#include "Vec3.h"
```

LeafBlockLoader.cpp の依存先関係図:

名前空間

- BCMFileIO

関数

- void BCMFileIO::DUMMY (void *)

6.19.1 詳解

LeafBlock ファイルを読み込むクラス

LeafBlockLoader.cpp に定義があります。

6.20 LeafBlockLoader.h ファイル

LeafBlock ファイルを読み込むクラス

```
#include <mpi.h>
#include "BCMFileCommon.h"
#include "IdxBlock.h"
#include "PartitionMapper.h"
#include "BlockManager.h"
#include "Scalar3D.h"
#include "Vec3.h"
```

LeafBlockLoader.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::LeafBlockLoader

  LeafBlock **ファイルを読み込むクラス**
- struct BCMFileIO::LeafBlockLoader::CellIDCapsule

  グリッドヘッダとデータを一括りにした構造体

名前空間

- BCMFileIO

6.20.1 詳解

LeafBlock ファイルを読み込むクラス

LeafBlockLoader.h に定義があります。

6.21 LeafBlockSaver.cpp ファイル

LeafBlock ファイルを出力する関数群

```
#include "LeafBlockSaver.h"
#include "BCMFileCommon.h"
#include "BitVoxel.h"
#include "BCMRLE.h"
#include "ErrorUtil.h"
#include "Logger.h"
#include "FileSystemUtil.h"
#include "BlockManager.h"
#include "Scalar3D.h"
#include "BCMTypes.h"
#include "Vec3.h"
```

LeafBlockSaver.cpp の依存先関係図:

名前空間

- BCMFileIO

6.21.1 詳解

LeafBlock ファイルを出力する関数群

LeafBlockSaver.cpp に定義があります。

6.22 LeafBlockSaver.h ファイル

LeafBlock ファイルを出力する関数群

```
#include <mpi.h>
#include "BCMFileCommon.h"
#include "IdxBlock.h"
#include "Vec3.h"
```

LeafBlockSaver.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::LeafBlockSaver

  LeafBlock **ファイルを出力する関数群**

名前空間

- BCMFileIO

6.22.1 詳解

LeafBlock ファイルを出力する関数群

LeafBlockSaver.h に定義があります。

6.23 Logger.cpp ファイル

ログ出力ユーティリティ

```
#include <stdio.h>
#include <stdarg.h>
#include <string>
#include <mpi.h>
#include "Logger.h"
```

Logger.cpp の依存先関係図:

名前空間

- BCMFileIO

6.23.1 詳解

ログ出力ユーティリティ

Logger.cpp に定義があります。

6.24 Logger.h ファイル

ログ出力ユーティリティ

被依存関係図:

クラス

- class BCMFileIO::Logger
  ログ出力ユーティリティ

名前空間

- BCMFileIO

6.24.1 詳解

ログ出力ユーティリティ

Logger.h に定義があります。

6.25 mpi_stubs.h ファイル

マクロ定義

- #define MPI_COMM_WORLD 0
- #define MPI_INT 1
- #define MPI_CHAR 2
- #define MPI_SUCCESS true

型定義

- typedef int MPI_Comm
- typedef int MPI_Datatype

関数

- bool MPI_Init (int *argc, char ***argv)
- int MPI_Comm_rank (MPI_Comm comm, int *rank)
- int MPI_Comm_size (MPI_Comm comm, int *size)
- int MPI_Allgather (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm)
- int MPI_Gather (void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvcnt, MPI_Datatype recvtype, int root, MPI_Comm comm)

6.25.1 マクロ定義詳解

6.25.1.1 #define MPI_CHAR 2

mpi_stubs.h の 21 行目に定義があります。

6.25.1.2 #define MPI_COMM_WORLD 0

mpi_stubs.h の 19 行目に定義があります。

参照元 BCMFileIO::BCMFileSaver::SaveIndexProc().

6.25.1.3 #define MPI_INT 1

mpi_stubs.h の 20 行目に定義があります。

6.25.1.4 #define MPI_SUCCESS true

mpi_stubs.h の 23 行目に定義があります。

6.25.2 型定義詳解

6.25.2.1 typedef int MPI_Comm

mpi_stubs.h の 17 行目に定義があります。

6.25.2.2 typedef int MPI_Datatype

mpi_stubs.h の 18 行目に定義があります。

6.25.3 関数詳解

6.25.3.1 int MPI_Allgather (void * *sendbuf*, int *sendcount*, MPI_Datatype *sendtype*, void * *recvbuf*, int *recvcount*, MPI_Datatype *recvtype*, MPI_Comm *comm*) [inline]

mpi_stubs.h の 39 行目に定義があります。

```
42 {
43   return 0;
44 }
```

6.25.3.2 int MPI_Comm_rank (MPI_Comm *comm*, int * *rank*) [inline]

mpi_stubs.h の 27 行目に定義があります。

参照元 BCMFileIO::BCMFileSaver::SaveIndexProc().

```
28 {
29   *rank = 0;
30   return 0;
31 }
```

6.25.3.3 int MPI_Comm_size (MPI_Comm *comm*, int * *size*) [inline]

mpi_stubs.h の 33 行目に定義があります。

参照元 BCMFileIO::BCMFileSaver::SaveIndexProc().

```
34 {
35   *size = 1;
36   return 0;
37 }
```

6.25.3.4 int MPI_Gather (void * *sendbuf*, int *sendcnt*, MPI_Datatype *sendtype*, void * *recvbuf*, int *recvcnt*, MPI_Datatype *recvtype*, int *root*, MPI_Comm *comm*) [inline]

mpi_stubs.h の 46 行目に定義があります。

```
49 {
50   return 0;
51 }
```

6.25.3.5 bool MPI_Init (int * *argc*, char *** *argv*) [inline]

mpi_stubs.h の 25 行目に定義があります。

```
25 { return true; }
```

6.26 PartitionMapper.h ファイル

MxN データロードのためのマッピングクラス

```
#include "Partition.h"
```

PartitionMapper.h の依存先関係図: 被依存関係図:

クラス

- class BCMFileIO::PartitionMapper
  MxN **データロードのためのマッピングクラス**
- struct BCMFileIO::PartitionMapper::FDIDList
  ファイル*ID* **とファイル内のデータ***ID* **リスト構造体**

名前空間

- BCMFileIO

6.26.1 詳解

MxN データロードのためのマッピングクラス 用語リスト

- did : グローバルなデータID
- FID : ファイルID
- FDID : ファイル内の相対データID

PartitionMapper.h に定義があります。

6.27 Vec3.h ファイル

Vec3<T> class Header.

```
#include <iostream>
#include <math.h>
```

Vec3.h の依存先関係図: 被依存関係図:

クラス

- class Vec3class::Vec3< T >

名前空間

- Vec3class

型定義

- typedef Vec3< unsigned char > Vec3class::Vec3uc
- typedef Vec3< int > Vec3class::Vec3i
- typedef Vec3< float > Vec3class::Vec3f
- typedef Vec3< double > Vec3class::Vec3d

列挙型

- enum Vec3class::AxisEnum { Vec3class::AXIS_X = 0, Vec3class::AXIS_Y, Vec3class::AXIS_Z, Vec3class::AXIS_ERROR }

関数

- template<typename T >
  Vec3< T > Vec3class::operator* (T s, const Vec3< T > &v)
- template<typename T >
  Vec3< T > Vec3class::multi (const Vec3< T > &a, const Vec3< T > &b)
- template<typename T >
  T Vec3class::dot (const Vec3< T > &a, const Vec3< T > &b)
- template<typename T >
  Vec3< T > Vec3class::cross (const Vec3< T > &a, const Vec3< T > &b)
- template<typename T >
  T Vec3class::distanceSquared (const Vec3< T > &a, const Vec3< T > &b)
- template<typename T >
  T Vec3class::distance (const Vec3< T > &a, const Vec3< T > &b)
- bool Vec3class::lessVec3f (const Vec3f &a, const Vec3f &b)
- template<typename T >
  std::istream & Vec3class::operator>> (std::istream &is, Vec3< T > &v)
- template<typename T >
  std::ostream & Vec3class::operator<< (std::ostream &os, const Vec3< T > &v)
- std::istream & Vec3class::operator>> (std::istream &is, Vec3uc &v)
- std::ostream & Vec3class::operator<< (std::ostream &os, const Vec3uc &v)

6.27.1 詳解

Vec3<T> class Header.

著者

aics

Vec3.h に定義があります。