Project 3.0

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Block Class Reference

```
#include <Block.h>
```

Public Member Functions

- Block ()
- std::string serialize () const

  Serializes the block to a string.
- void dump () const

  Dumps the block content to standard output.

Static Public Member Functions

- static Block deserialize (const std::string &data)

  Deserializes a block from a string.

Public Attributes

- int blockNumber

  Sequential number of the block.
- int nextBlock

  Logical pointer to the next block (-1 if none).
- std::vector< Record > records

  List of records in this block.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Block()

```
Block::Block ( )  [inline]
```

3.1.2 Member Function Documentation

3.1.2.1 deserialize()

```
static Block Block::deserialize (
             const std::string & data )  [inline], [static]
```

Deserializes a block from a string.

Expects the first line to be the block header. Each subsequent line is a packed record.

Parameters

data	The serialized block string.

Returns

A Block object.

3.1.2.2 dump()

```
void Block::dump ( ) const  [inline]
```

Dumps the block content to standard output.

3.1.2.3 serialize()

```
std::string Block::serialize ( ) const  [inline]
```

Serializes the block to a string.

First writes a header line: blockNumber,recordCount,nextBlock Then, for each record, packs the record using Buffer and writes the result.

Returns

The serialized block string.

3.1.3 Member Data Documentation

3.1.3.1 blockNumber

```
int Block::blockNumber
```

Sequential number of the block.

3.1.3.2 nextBlock

```
int Block::nextBlock
```

Logical pointer to the next block (-1 if none).

3.1.3.3 records

```
std::vector<Record> Block::records
```

List of records in this block.

The documentation for this class was generated from the following file:

- Block.h

3.2 BlockBuffer Class Reference

```
#include <BlockBuffer.h>
```

Public Member Functions

- bool writeBlocks (const std::string &filename, const std::vector< Block > &blocks)

  Writes a blocked sequence set file.
- bool readBlocks (const std::string &filename, std::vector< Block > &blocks)

  Reads a blocked sequence set file.

3.2.1 Member Function Documentation

3.2.1.1 readBlocks()

```
bool BlockBuffer::readBlocks (
            const std::string & filename,
            std::vector< Block > & blocks )
```

Reads a blocked sequence set file.

Parameters

filename	The input file name.
blocks	A vector to receive the blocks.

Returns

true on success.

3.2.1.2 writeBlocks()

```
bool BlockBuffer::writeBlocks (
            const std::string & filename,
            const std::vector< Block > & blocks )
```

Writes a blocked sequence set file.

The file consists of:

- A file header (packed using Buffer)
- A line with the number of blocks
- For each block: a length indicator (the size of the packed block) and the packed block data.

Parameters

filename	The output file name.
blocks	A vector of blocks to write.

Returns

true on success.

The documentation for this class was generated from the following files:

- BlockBuffer.h
- BlockBuffer.cpp

3.3 Buffer Class Reference

```
#include <Buffer.h>
```

Public Member Functions

- void pack (const std::string &data)

  Packs a string into a length-indicated format.
- std::string unpack ()

  Unpacks the string (ignores the length indicator).
- void readHeader (std::ifstream &file)

  Reads the header record from the input file stream.
- void writeHeader (std::ofstream &file)

  Writes the header record to the output file stream.
- std::string getBuffer () const

  Returns the internal packed string.

3.3.1 Member Function Documentation

3.3.1.1 getBuffer()

```
std::string Buffer::getBuffer ( ) const  [inline]
```

Returns the internal packed string.

Returns

The packed string.

3.3.1.2 pack()

```
void Buffer::pack (
             const std::string & data )
```

Packs a string into a length-indicated format.

Example: "Hello" becomes "5,Hello"

Parameters

data	The string to pack.

3.3.1.3 readHeader()

```
void Buffer::readHeader (
             std::ifstream & file )
```

Reads the header record from the input file stream.

Parameters

file	The input stream.

3.3.1.4 unpack()

```
std::string Buffer::unpack ( )
```

Unpacks the string (ignores the length indicator).

Unpacks a length-indicated string from the buffer.

Returns

The original string.

3.3.1.5 writeHeader()

```
void Buffer::writeHeader (
             std::ofstream & file )
```

Writes the header record to the output file stream.

Parameters

file	The output stream.

The documentation for this class was generated from the following files:

- Buffer.h
- Buffer.cpp

3.4 Record Class Reference

```
#include <Record.h>
```

Public Member Functions

- Record ()
- std::string serialize () const

    Serializes the record as a CSV string.

Static Public Member Functions

- static Record deserialize (const std::string &data)

    Deserializes a CSV string into a Record.

Public Attributes

- int index
- std::string field1
- std::string field2
- std::string field3

3.4.1 Constructor & Destructor Documentation

3.4.1.1 Record()

```
Record::Record ( )  [inline]
```

3.4.2 Member Function Documentation

3.4.2.1 deserialize()

```
static Record Record::deserialize (
             const std::string & data )  [inline], [static]
```

Deserializes a CSV string into a Record.

Parameters

data	The CSV string.

Returns

A Record object.

3.4.2.2 serialize()

```
std::string Record::serialize ( ) const  [inline]
```

Serializes the record as a CSV string.

Format: index,field1,field2,field3

3.4.3 Member Data Documentation

3.4.3.1 field1

```
std::string Record::field1
```

3.4.3.2 field2

```
std::string Record::field2
```

3.4.3.3 field3

```
std::string Record::field3
```

3.4.3.4 index

```
int Record::index
```

The documentation for this class was generated from the following file:

- Record.h

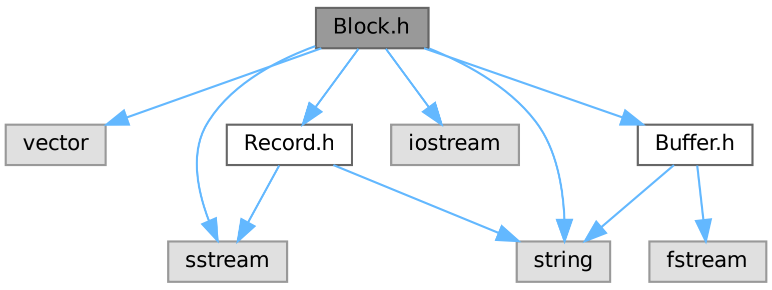
Chapter 4

File Documentation

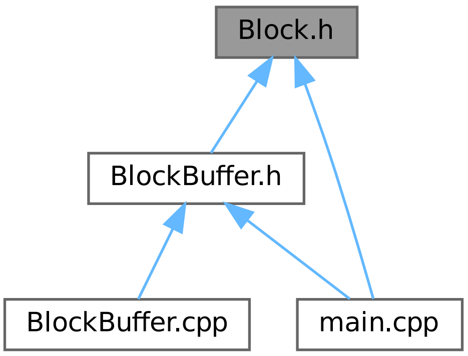
4.1 Block.h File Reference

```
#include <vector>
#include <string>
#include <sstream>
#include <iostream>
#include "Record.h"
#include "Buffer.h"
```

Include dependency graph for Block.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Block

4.2 Block.h

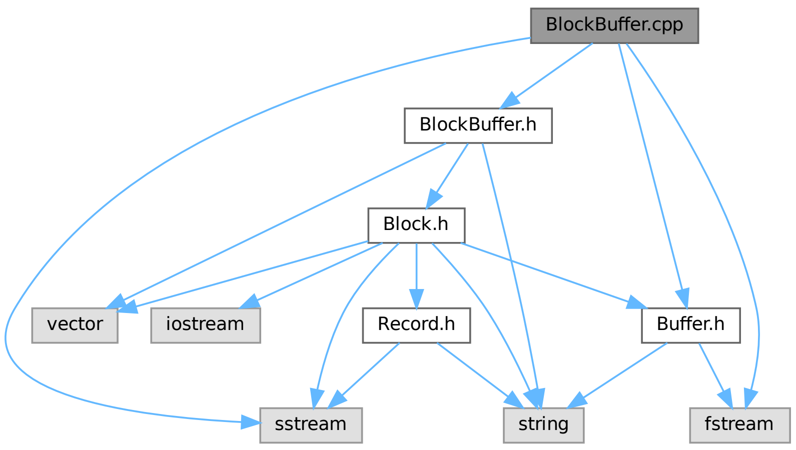
Go to the documentation of this file.

```
#ifndef BLOCK_H
#define BLOCK_H

#include <vector>
#include <string>
#include <sstream>
#include <iostream>
#include "Record.h"
#include "Buffer.h"

class Block {
public:
    int blockNumber;
    int nextBlock;
    std::vector<Record> records;

    Block() : blockNumber(0), nextBlock(-1) {}

    std::string serialize() const {
        std::stringstream ss;
        // Write block header.
        ss « blockNumber « "," « records.size() « "," « nextBlock « "\n";
        // Write each record (packed with Buffer).
        for (const auto &rec : records) {
            Buffer buf;
            std::string recStr = rec.serialize();
            buf.pack(recStr);
            ss « buf.getBuffer() « "\n";
        }
        return ss.str();
    }

    static Block deserialize(const std::string &data) {
        Block blk;
        std::stringstream ss(data);
        std::string line;
        // Get header line.
        if (getline(ss, line)) {
            std::stringstream headerStream(line);
            std::string token;
            getline(headerStream, token, ',');
            blk.blockNumber = std::stoi(token);
            getline(headerStream, token, ','); // record count (not used here)
            int recordCount = std::stoi(token);
            getline(headerStream, token, ',');
            blk.nextBlock = std::stoi(token);
        }
        // Read each packed record.
        while (getline(ss, line)) {
            if (line.empty())
                continue;
            // Unpack the record manually.
            size_t commaPos = line.find(',');
            if (commaPos == std::string::npos)
                continue;
            int len = std::stoi(line.substr(0, commaPos));
            std::string recData = line.substr(commaPos + 1, len);
            Record r = Record::deserialize(recData);
            blk.records.push_back(r);
        }
        return blk;
    }

    void dump() const {
        std::cout « "Block Number: " « blockNumber « ", Next Block: " « nextBlock « std::endl;
        std::cout « "Records:" « std::endl;
        for (const auto &r : records) {
            std::cout « r.index « " | " « r.field1 « " | " « r.field2 « " | " « r.field3 « std::endl;
        }
    }
};

#endif
```

4.3 BlockBuffer.cpp File Reference

```
#include "BlockBuffer.h"
#include "Buffer.h"
#include <fstream>
#include <sstream>
```

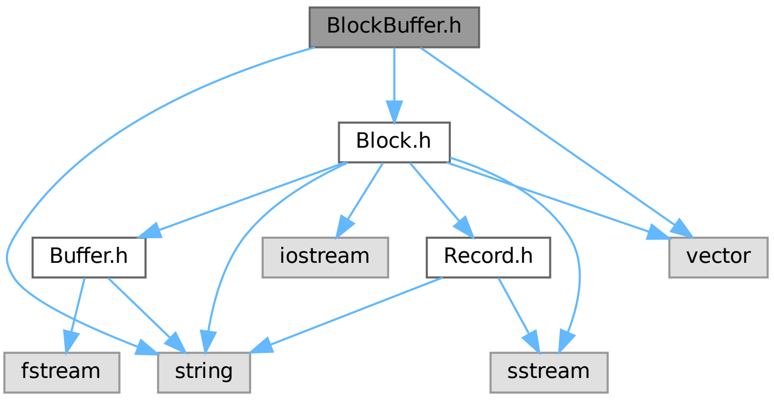
Include dependency graph for BlockBuffer.cpp:



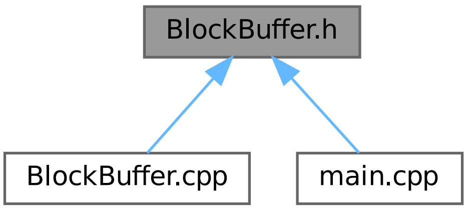
4.4 BlockBuffer.h File Reference

```
#include <string>
#include <vector>
#include "Block.h"
```

Include dependency graph for BlockBuffer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BlockBuffer

4.5 BlockBuffer.h

Go to the documentation of this file.

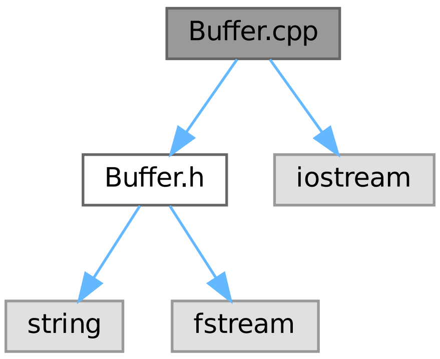
```
#ifndef BLOCKBUFFER_H
#define BLOCKBUFFER_H

#include <string>
```

```
#include <vector>
#include "Block.h"

class BlockBuffer {
public:
    bool writeBlocks(const std::string &filename, const std::vector<Block> &blocks);

    bool readBlocks(const std::string &filename, std::vector<Block> &blocks);
};

#endif
```

4.6 Buffer.cpp File Reference

```
#include "Buffer.h"
#include <iostream>
```

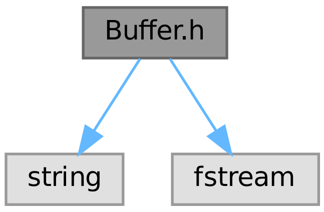
Include dependency graph for Buffer.cpp:



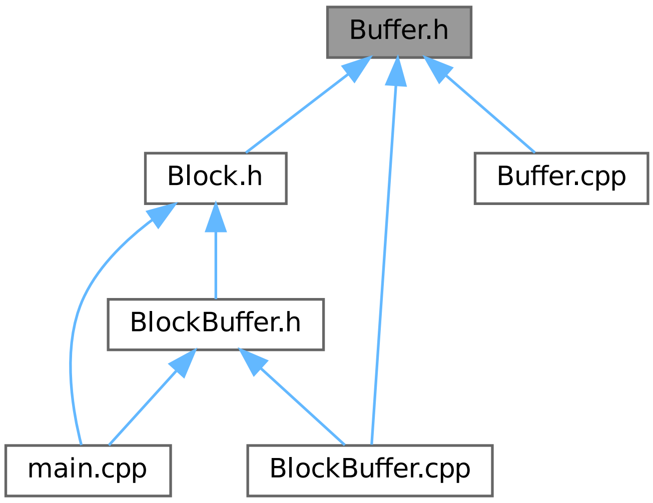
4.7 Buffer.h File Reference

```
#include <string>
#include <fstream>
```

Include dependency graph for Buffer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Buffer

4.8 Buffer.h

Go to the documentation of this file.

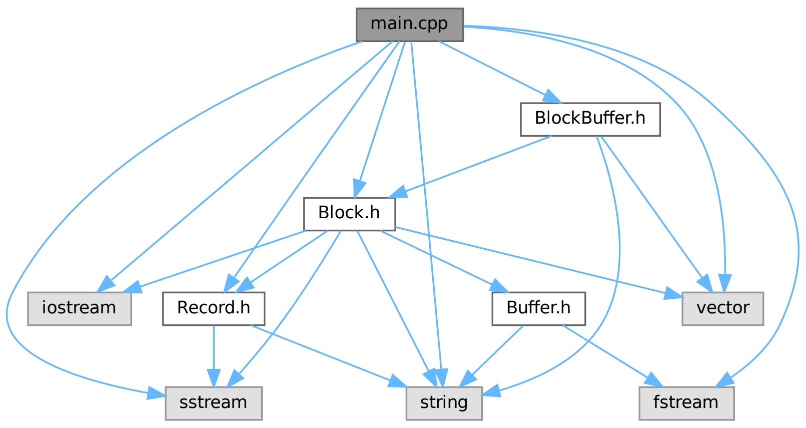
```
#ifndef BUFFER_H
#define BUFFER_H

#include <string>
#include <fstream>

class Buffer {
public:
    void pack(const std::string& data);

    std::string unpack();

    void readHeader(std::ifstream& file);

    void writeHeader(std::ofstream& file);

    std::string getBuffer() const { return buffer; }

private:
    std::string buffer;
};

#endif
```

4.9 main.cpp File Reference

```
#include <iostream>
#include <fstream>
```

```
#include <sstream>
#include <vector>
#include <string>
#include "BlockBuffer.h"
#include "Block.h"
#include "Record.h"
```

Include dependency graph for main.cpp:



Functions

- vector< string > readCSV (const string &filename)

  Reads a CSV file (with a header) and returns a vector of CSV record strings.
- vector< Block > createBlocks (const vector< string > &records, int recordsPerBlock)

  Creates blocks from CSV record strings.
- void dumpPhysical (const vector< Block > &blocks)

  Dump blocks in physical order (as stored in file).
- void dumpLogical (const vector< Block > &blocks)

  Dump blocks in logical order (following nextBlock pointer).
- int main (int argc, char *argv[ ])

4.9.1 Function Documentation

4.9.1.1 createBlocks()

```
vector< Block > createBlocks (
            const vector< string > & records,
            int recordsPerBlock )
```

Creates blocks from CSV record strings.

4.9.1.2 dumpLogical()

```
void dumpLogical (
            const vector< Block > & blocks )
```

Dump blocks in logical order (following nextBlock pointer).

4.9.1.3 dumpPhysical()

```
void dumpPhysical (
            const vector< Block > & blocks )
```

Dump blocks in physical order (as stored in file).

4.9.1.4 main()

```
int main (
            int argc,
            char * argv[] )
```

4.9.1.5 readCSV()

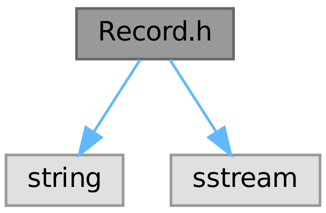
```
vector< string > readCSV (
            const string & filename )
```

Reads a CSV file (with a header) and returns a vector of CSV record strings.

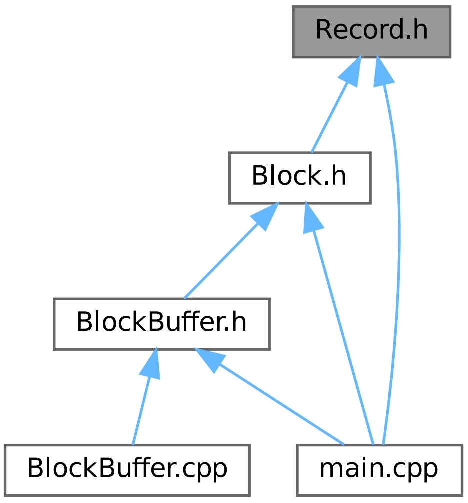
4.10 Record.h File Reference

```
#include <string>
#include <sstream>
```

Include dependency graph for Record.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Record

4.11 Record.h

Go to the documentation of this file.

```
#ifndef RECORD_H
#define RECORD_H

#include <string>
#include <sstream>

class Record {
public:
    int index;
    std::string field1;
    std::string field2;
    std::string field3;

    Record() : index(0) {}

    std::string serialize() const {
        std::stringstream ss;
        ss « index « "," « field1 « "," « field2 « "," « field3;
        return ss.str();
    }

    static Record deserialize(const std::string &data) {
        Record r;
        std::stringstream ss(data);
        std::string token;
        getline(ss, token, ',');
        r.index = std::stoi(token);
        getline(ss, r.field1, ',');
        getline(ss, r.field2, ',');
        getline(ss, r.field3, ',');
        return r;
    }
};

#endif
```

Index