

Project 3.0

Generated by Doxygen 1.9.8

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Block Class Reference	5
3.1.1 Constructor & Destructor Documentation	5
3.1.1.1 Block()	5
3.1.2 Member Function Documentation	6
3.1.2.1 deserialize()	6
3.1.2.2 dump()	6
3.1.2.3 serialize()	6
3.1.3 Member Data Documentation	6
3.1.3.1 blockNumber	6
3.1.3.2 nextBlock	7
3.1.3.3 records	7
3.2 BlockBuffer Class Reference	7
3.2.1 Member Function Documentation	7
3.2.1.1 readBlocks()	7
3.2.1.2 writeBlocks()	8
3.3 Buffer Class Reference	8
3.3.1 Member Function Documentation	9
3.3.1.1 getBuffer()	9
3.3.1.2 pack()	9
3.3.1.3 readHeader()	9
3.3.1.4 unpack()	9
3.3.1.5 writeHeader()	10
3.4 Record Class Reference	10
3.4.1 Constructor & Destructor Documentation	10
3.4.1.1 Record()	10
3.4.2 Member Function Documentation	10
3.4.2.1 deserialize()	10
3.4.2.2 serialize()	11
3.4.3 Member Data Documentation	11
3.4.3.1 field1	11
3.4.3.2 field2	11
3.4.3.3 field3	11
3.4.3.4 index	11
4 File Documentation	13
4.1 Block.h File Reference	13

4.2 Block.h	14
4.3 BlockBuffer.cpp File Reference	15
4.4 BlockBuffer.h File Reference	16
4.5 BlockBuffer.h	16
4.6 Buffer.cpp File Reference	17
4.7 Buffer.h File Reference	17
4.8 Buffer.h	18
4.9 main.cpp File Reference	18
4.9.1 Function Documentation	19
4.9.1.1 createBlocks()	19
4.9.1.2 dumpLogical()	19
4.9.1.3 dumpPhysical()	20
4.9.1.4 main()	20
4.9.1.5 readCSV()	20
4.10 Record.h File Reference	20
4.11 Record.h	21
Index	23

Chapter 1

Class Index

1.1 Class List

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

Block	5
BlockBuffer	7
Buffer	8
Record	10

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Block.h	13
BlockBuffer.cpp	15
BlockBuffer.h	16
Buffer.cpp	17
Buffer.h	17
main.cpp	18
Record.h	20

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

<i>data</i>	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

<i>filename</i>	The input file name.
<i>blocks</i>	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

<i>filename</i>	The output file name.
<i>blocks</i>	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

<i>data</i>	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

<i>file</i>	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

<i>file</i>	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

<i>data</i>	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](#)

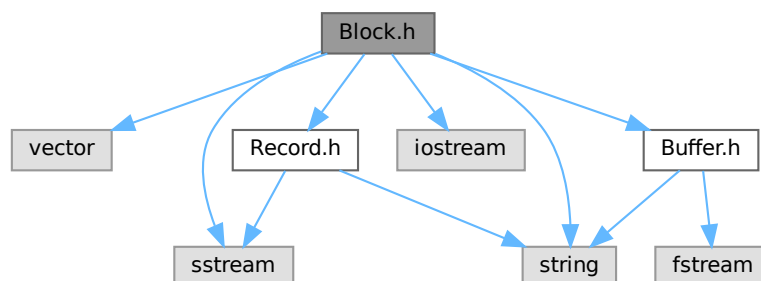
Chapter 4

File Documentation

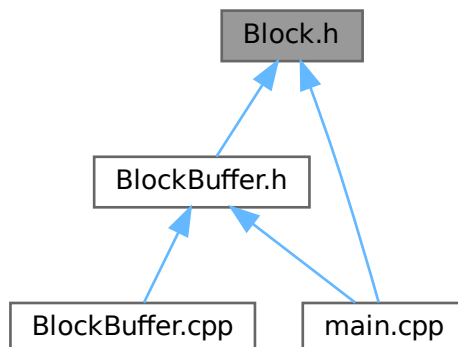
4.1 Block.h File Reference

```
#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.](#)

```

00001 #ifndef BLOCK_H
00002 #define BLOCK_H
00003
00004 #include <vector>
00005 #include <string>
00006 #include <sstream>
00007 #include <iostream>
00008 #include "Record.h"
00009 #include "Buffer.h"
00010
00011 class Block {
00012 public:
00013     int blockNumber;
00014     int nextBlock;
00015     std::vector<Record> records;
00016
00017     Block() : blockNumber(0), nextBlock(-1) {}
00018
00019     std::string serialize() const {
00020         std::stringstream ss;
00021         // Write block header.
00022         ss << blockNumber << "," << records.size() << "," << nextBlock << "\n";
00023         // Write each record (packed with Buffer).
00024         for (const auto &rec : records) {
00025             Buffer buf;
00026             std::string recStr = rec.serialize();
00027             buf.pack(recStr);
00028             ss << buf.getBuffer() << "\n";
00029         }
00030         return ss.str();
00031     }
00032
00033     static Block deserialize(const std::string &data) {
00034         Block blk;
00035         std::stringstream ss(data);
00036         std::string line;
00037         // Get header line.
00038         if (getline(ss, line)) {
00039             std::stringstream headerStream(line);

```

```

00057         std::string token;
00058         getline(headerStream, token, ',');
00059         blk.blockNumber = std::stoi(token);
00060         getline(headerStream, token, ','); // record count (not used here)
00061         int recordCount = std::stoi(token);
00062         getline(headerStream, token, ',');
00063         blk.nextBlock = std::stoi(token);
00064     }
00065     // Read each packed record.
00066     while (getline(ss, line)) {
00067         if (line.empty())
00068             continue;
00069         // Unpack the record manually.
00070         size_t commaPos = line.find(',');
00071         if (commaPos == std::string::npos)
00072             continue;
00073         int len = std::stoi(line.substr(0, commaPos));
00074         std::string recData = line.substr(commaPos + 1, len);
00075         Record r = Record::deserialize(recData);
00076         blk.records.push_back(r);
00077     }
00078     return blk;
00079 }
00080
00084 void dump() const {
00085     std::cout << "Block Number: " << blockNumber << ", Next Block: " << nextBlock << std::endl;
00086     std::cout << "Records:" << std::endl;
00087     for (const auto &r : records) {
00088         std::cout << r.index << " | " << r.field1 << " | " << r.field2 << " | " << r.field3 << std::endl;
00089     }
00090 }
00091 };
00092
00093 #endif

```

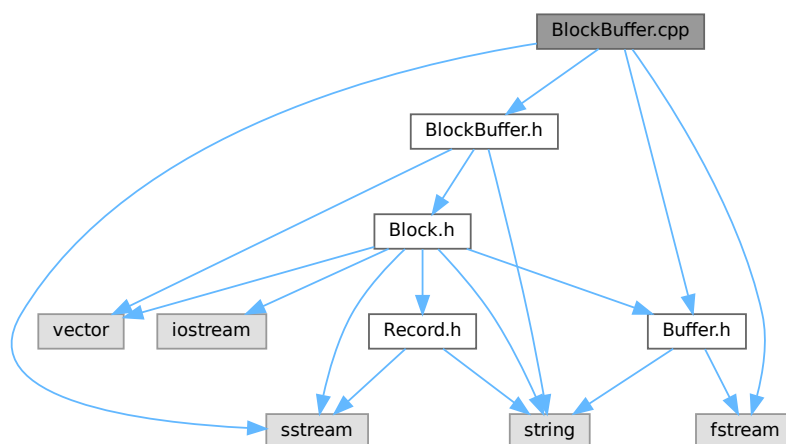
4.3 BlockBuffer.cpp File Reference

```

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

```

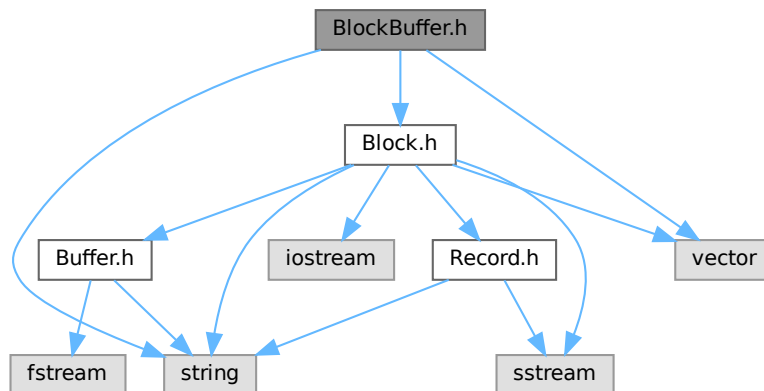
Include dependency graph for BlockBuffer.cpp:



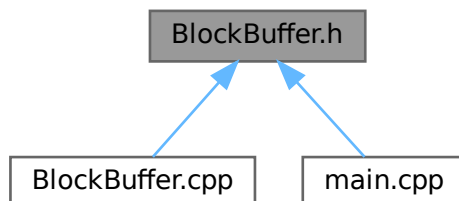
4.4 BlockBuffer.h File Reference

```
#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.](#)

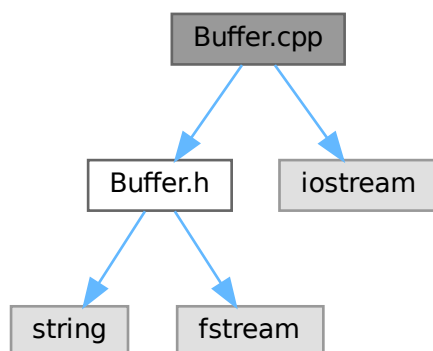
```
00001 #ifndef BLOCKBUFFER_H
00002 #define BLOCKBUFFER_H
00003
00004 #include <string>
```

```
00005 #include <vector>
00006 #include "Block.h"
00007
00008 class BlockBuffer {
00009 public:
00022     bool writeBlocks(const std::string &filename, const std::vector<Block> &blocks);
00023
00031     bool readBlocks(const std::string &filename, std::vector<Block> &blocks);
00032 };
00033
00034 #endif
```

4.6 Buffer.cpp File Reference

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

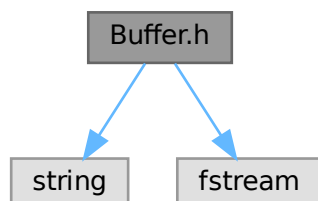
Include dependency graph for Buffer.cpp:



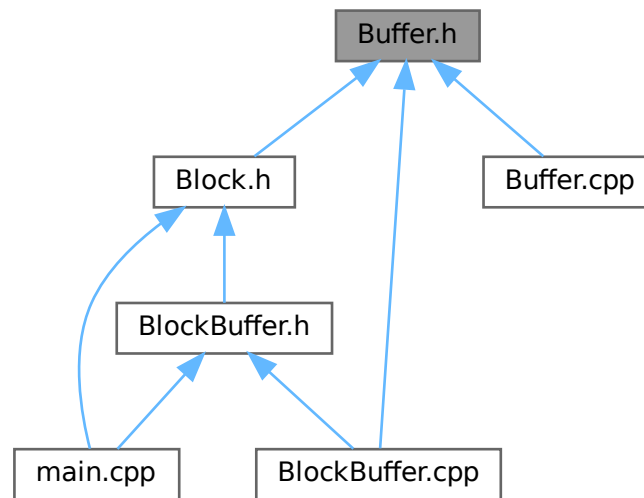
4.7 Buffer.h File Reference

```
#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.](#)

```

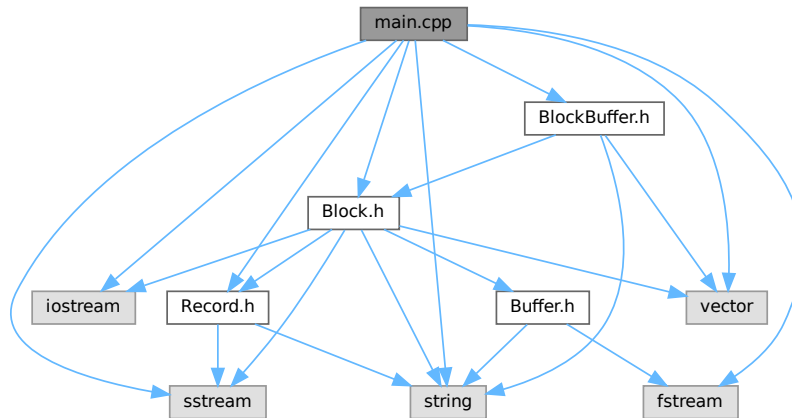
00001 #ifndef BUFFER_H
00002 #define BUFFER_H
00003
00004 #include <string>
00005 #include <fstream>
00006
00007 class Buffer {
00008 public:
00016     void pack(const std::string& data);
00017
00023     std::string unpack();
00024
00030     void readHeader(std::ifstream& file);
00031
00037     void writeHeader(std::ofstream& file);
00038
00044     std::string getBuffer() const { return buffer; }
00045
00046 private:
00047     std::string buffer;
00048 };
00049
00050 #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()

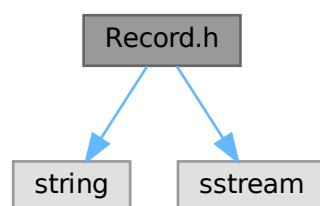
```
vector< string > readCSV (
    const string & filename )
```

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

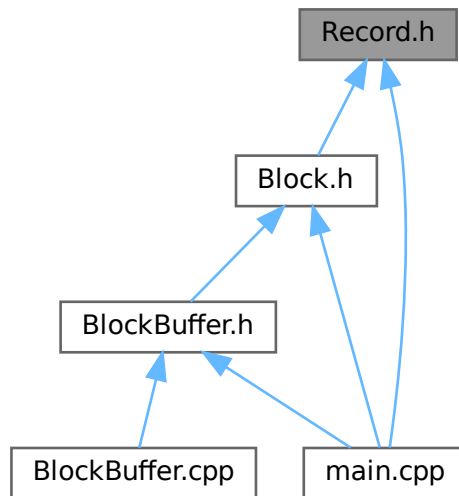
4.10 Record.h File Reference

```
#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.](#)

```

00001 #ifndef RECORD_H
00002 #define RECORD_H
00003
00004 #include <string>
00005 #include <sstream>
00006
00007 class Record {
00008 public:
00009     int index;
00010     std::string field1;
00011     std::string field2;
00012     std::string field3;
00013
00014     Record() : index(0) {}
00015
00021     std::string serialize() const {
00022         std::stringstream ss;
00023         ss << index << "," << field1 << "," << field2 << "," << field3;
00024         return ss.str();
00025     }
00026
00033     static Record deserialize(const std::string &data) {
00034         Record r;
00035         std::stringstream ss(data);
00036         std::string token;
00037         getline(ss, token, ',');
00038         r.index = std::stoi(token);
00039         getline(ss, r.field1, ',');
00040         getline(ss, r.field2, ',');
00041         getline(ss, r.field3, ',');
00042         return r;
00043     }
00044 };
00045
00046 #endif
  
```


Index

- Block, [5](#)
 - Block, [5](#)
 - blockNumber, [6](#)
 - deserialize, [6](#)
 - dump, [6](#)
 - nextBlock, [6](#)
 - records, [7](#)
 - serialize, [6](#)
- Block.h, [13](#)
- BlockBuffer, [7](#)
 - readBlocks, [7](#)
 - writeBlocks, [7](#)
- BlockBuffer.cpp, [15](#)
- BlockBuffer.h, [16](#)
- blockNumber
 - Block, [6](#)
- Buffer, [8](#)
 - getBuffer, [9](#)
 - pack, [9](#)
 - readHeader, [9](#)
 - unpack, [9](#)
 - writeHeader, [9](#)
- Buffer.cpp, [17](#)
- Buffer.h, [17](#)
- createBlocks
 - main.cpp, [19](#)
- deserialize
 - Block, [6](#)
 - Record, [10](#)
- dump
 - Block, [6](#)
- dumpLogical
 - main.cpp, [19](#)
- dumpPhysical
 - main.cpp, [19](#)
- field1
 - Record, [11](#)
- field2
 - Record, [11](#)
- field3
 - Record, [11](#)
- getBuffer
 - Buffer, [9](#)
- index
 - Record, [11](#)
- main
 - main.cpp, [20](#)
- main.cpp, [18](#)
 - createBlocks, [19](#)
 - dumpLogical, [19](#)
 - dumpPhysical, [19](#)
 - main, [20](#)
 - readCSV, [20](#)
- nextBlock
 - Block, [6](#)
- pack
 - Buffer, [9](#)
- readBlocks
 - BlockBuffer, [7](#)
- readCSV
 - main.cpp, [20](#)
- readHeader
 - Buffer, [9](#)
- Record, [10](#)
 - deserialize, [10](#)
 - field1, [11](#)
 - field2, [11](#)
 - field3, [11](#)
 - index, [11](#)
 - Record, [10](#)
 - serialize, [11](#)
- Record.h, [20](#)
- records
 - Block, [7](#)
- serialize
 - Block, [6](#)
 - Record, [11](#)
- unpack
 - Buffer, [9](#)
- writeBlocks
 - BlockBuffer, [7](#)
- writeHeader
 - Buffer, [9](#)