

## BLINK DB 24CS60R02 - PART B

Generated by Doxygen 1.13.2



|  |           |
|--|-----------|
| <b>1 Class Index</b>                               | <b>1</b>  |
| 1.1 Class List                                     | 1         |
| <b>2 File Index</b>                                | <b>3</b>  |
| 2.1 File List                                      | 3         |
| <b>3 Class Documentation</b>                       | <b>5</b>  |
| 3.1 BenchmarkData Class Reference                  | 5         |
| 3.1.1 Detailed Description                         | 6         |
| 3.1.2 Constructor & Destructor Documentation       | 6         |
| 3.1.2.1 BenchmarkData()                            | 6         |
| 3.1.3 Member Function Documentation                | 6         |
| 3.1.3.1 generateRandomString()                     | 6         |
| 3.1.3.2 generateTestData()                         | 7         |
| 3.2 KQueueServer Class Reference                   | 7         |
| 3.2.1 Detailed Description                         | 8         |
| 3.2.2 Constructor & Destructor Documentation       | 8         |
| 3.2.2.1 KQueueServer()                             | 8         |
| 3.2.2.2 ~KQueueServer()                            | 8         |
| 3.2.3 Member Function Documentation                | 8         |
| 3.2.3.1 handleClient()                             | 8         |
| 3.2.3.2 processCommand()                           | 9         |
| 3.2.3.3 run()                                      | 9         |
| 3.3 RespParser Class Reference                     | 10        |
| 3.3.1 Detailed Description                         | 10        |
| 3.3.2 Member Function Documentation                | 10        |
| 3.3.2.1 createError()                              | 10        |
| 3.3.2.2 createSimpleString()                       | 11        |
| 3.3.2.3 parseArray()                               | 12        |
| 3.3.2.4 serializeBulkString()                      | 12        |
| <b>4 File Documentation</b>                        | <b>15</b> |
| 4.1 benchmarkdata/benchmarkdata.cpp File Reference | 15        |
| 4.1.1 Detailed Description                         | 15        |
| 4.2 benchmarkdata/benchmarkdata.h File Reference   | 15        |
| 4.2.1 Detailed Description                         | 15        |
| 4.3 benchmarkdata.h                                | 16        |
| 4.4 main.cpp File Reference                        | 16        |
| 4.4.1 Detailed Description                         | 16        |
| 4.5 server/resp_parser.cpp File Reference          | 16        |
| 4.5.1 Detailed Description                         | 17        |
| 4.6 server/resp_parser.h File Reference            | 17        |
| 4.6.1 Detailed Description                         | 17        |

|  |           |
|--|-----------|
| 4.7 resp_parser.h . . . . .                    | 17        |
| 4.8 server/server.cpp File Reference . . . . . | 17        |
| 4.8.1 Detailed Description . . . . .           | 18        |
| 4.9 server/server.h File Reference . . . . .   | 18        |
| 4.9.1 Detailed Description . . . . .           | 18        |
| 4.10 server.h . . . . .                        | 18        |
| <b>Index</b>                                   | <b>19</b> |

# Chapter 1

## Class Index

### 1.1 Class List

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

|   |    |
|---|----|
| <a href="#">BenchmarkData</a>                                     |    |
| Generates and stores test data for benchmarking . . . . .         | 5  |
| <a href="#">KQueueServer</a>                                      |    |
| Server implementation using kqueue for I/O multiplexing . . . . . | 7  |
| <a href="#">RespParser</a>  |    |
| Parser for Redis RESP-2 protocol . . . . .                        | 10 |



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

|  |   |    |
|--|---|----|
| <a href="#">main.cpp</a>                         | Main entry point for the BLINK DB server application . . . . .                          | 16 |
| benchmarkdata/ <a href="#">benchmarkdata.cpp</a> | Implementation of benchmark data generator . . . . .                                    | 15 |
| benchmarkdata/ <a href="#">benchmarkdata.h</a>   | Benchmark data generator for LSM Tree testing . . . . .                                 | 15 |
| server/ <a href="#">resp_parser.cpp</a>          | Implementation of Redis RESP-2 protocol parser . . . . .                                | 16 |
| server/ <a href="#">resp_parser.h</a>            | Redis RESP-2 protocol parser implementation header file . . . . .                       | 17 |
| server/ <a href="#">server.cpp</a>               | Implementation of KQueue-based server for handling key-value store operations . . . . . | 17 |
| server/ <a href="#">server.h</a>                 | KQueue-based server for LSM Tree storage engine . . . . .                               | 18 |





## Chapter 3

# Class Documentation

### 3.1 BenchmarkData Class Reference

Generates and stores test data for benchmarking.

```
#include <benchmarkdata.h>
```

#### Public Member Functions

- [BenchmarkData](#) (size\_t reads=1000000, size\_t writes=1000000, size\_t keyLength=16, size\_t valueLength=16)  
*Constructor.*

#### Public Attributes

- std::vector< std::string > **keys**  
*Pre-generated keys.*
- std::vector< std::string > **values**  
*Pre-generated values.*

#### Private Member Functions

- std::string [generateRandomString](#) (size\_t length)  
*Generates a random string of specified length.*
- void [generateTestData](#) ()  
*Generates all test data (keys and values)*

#### Private Attributes

- size\_t **numReads**  
*Number of read operations to generate data for.*
- size\_t **numWrites**  
*Number of write operations to generate data for.*
- size\_t **keySize**  
*Size of generated keys in characters.*
- size\_t **valueSize**  
*Size of generated values in characters.*

### 3.1.1 Detailed Description

Generates and stores test data for benchmarking.

This class pre-generates random keys and values for consistent benchmark testing of storage engines.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 BenchmarkData()

```
BenchmarkData::BenchmarkData (
    size_t reads = 1000000,
    size_t writes = 1000000,
    size_t keyLength = 16,
    size_t valueLength = 16)
```

Constructor.

Constructs a [BenchmarkData](#) object with the given parameters.

##### Parameters

|                    |   |
|--------------------|---|
| <i>reads</i>       | Number of read operations (default: 1,000,000)  |
| <i>writes</i>      | Number of write operations (default: 1,000,000) |
| <i>keyLength</i>   | Length of each key (default: 16)                |
| <i>valueLength</i> | Length of each value (default: 16)              |

Initializes the number of read and write operations, key and value sizes, and generates the necessary test data.

##### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <i>reads</i>       | Number of read operations.  |
| <i>writes</i>      | Number of write operations. |
| <i>keyLength</i>   | Length of each key.         |
| <i>valueLength</i> | Length of each value.       |

### 3.1.3 Member Function Documentation

#### 3.1.3.1 generateRandomString()

```
std::string BenchmarkData::generateRandomString (
    size_t length) [private]
```

Generates a random string of specified length.

Generates a random alphanumeric string of a specified length.

**Parameters**

|               |                                      |
|---------------|--------------------------------------|
| <i>length</i> | The length of the string to generate |
|---------------|--------------------------------------|

**Returns**

A random string

Uses a random number generator to create a string consisting of uppercase letters, lowercase letters, and digits.

**Parameters**

|               |                                   |
|---------------|-----------------------------------|
| <i>length</i> | Length of the string to generate. |
|---------------|-----------------------------------|

**Returns**

A randomly generated string of the specified length.

**3.1.3.2 generateTestData()**

```
void BenchmarkData::generateTestData () [private]
```

Generates all test data (keys and values)

Generates test data consisting of random keys and values.

Populates the `keys` vector with randomly generated keys and the `values` vector with randomly generated values.

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

- benchmarkdata/[benchmarkdata.h](#)
- benchmarkdata/[benchmarkdata.cpp](#)

**3.2 KQueueServer Class Reference**

Server implementation using kqueue for I/O multiplexing.

```
#include <server.h>
```

**Public Member Functions**

- [KQueueServer](#) (LSMTree &store, [BenchmarkData](#) &data)  
*Constructor.*
- [~KQueueServer](#) ()  
*Destructor.*
- int [run](#) ()  
*Initializes and starts the server.*

### Private Member Functions

- `std::string processCommand` (const `std::vector< std::string >` &args, int rn)  
*Processes a command from a client.*
- `void handleClient` (int fd, int rn)  
*Handles a client connection.*

### Private Attributes

- `int server_fd`
- `int kq`
- `LSMTree & store`
- `BenchmarkData & data`

## 3.2.1 Detailed Description

Server implementation using kqueue for I/O multiplexing.

Handles client connections and processes Redis-compatible commands using the LSM Tree storage engine.

## 3.2.2 Constructor & Destructor Documentation

### 3.2.2.1 KQueueServer()

```
KQueueServer::KQueueServer (
    LSMTree & store,
    BenchmarkData & data)
```

Constructor.

Constructs a `KQueueServer` object.

#### Parameters

|              |  |
|--------------|--|
| <i>store</i> | LSM Tree storage engine                                |
| <i>data</i>  | Benchmark data   |
| <i>store</i> | Reference to the LSMTree storage engine.               |
| <i>data</i>  | Reference to the <code>BenchmarkData</code> generator. |

### 3.2.2.2 ~KQueueServer()

```
KQueueServer::~KQueueServer ()
```

Destructor.

Destroys the `KQueueServer` object and closes open file descriptors.

## 3.2.3 Member Function Documentation

### 3.2.3.1 handleClient()

```
void KQueueServer::handleClient (
    int fd,
    int rn) [private]
```

Handles a client connection.

Handles client requests.

## Parameters

|           |   |
|-----------|---|
| <i>fd</i> | Client socket file descriptor           |
| <i>rn</i> | Random number for benchmark data access |
| <i>fd</i> | Client socket file descriptor.          |
| <i>rn</i> | Random index for benchmark data access. |

**3.2.3.2 processCommand()**

```
std::string KQueueServer::processCommand (  
    const std::vector< std::string > & args,  
    int rn) [private]
```

Processes a command from a client.

Processes a client command and generates a response.

## Parameters

|             |   |
|-------------|---|
| <i>args</i> | Command arguments                       |
| <i>rn</i>   | Random number for benchmark data access |

## Returns

Response to send to client

## Parameters

|             |   |
|-------------|---|
| <i>args</i> | Parsed command arguments.               |
| <i>rn</i>   | Random index for benchmark data access. |

## Returns

A RESP-formatted response string.

**3.2.3.3 run()**

```
int KQueueServer::run ()
```

Initializes and starts the server.

Runs the KQueue-based event-driven server.

## Returns

0 on success, error code otherwise  
0 on successful execution, 1 on error.

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

- [server/server.h](#)
- [server/server.cpp](#)

### 3.3 RespParser Class Reference

Parser for Redis RESP-2 protocol.

```
#include <resp_parser.h>
```

#### Static Public Member Functions

- static std::vector< std::string > [parseArray](#) (const std::string &buffer)  
*Parses a RESP array message into command arguments.*
- static std::string [serializeBulkString](#) (const std::string &value)  
*Serializes a string into RESP bulk string format.*
- static std::string [createSimpleString](#) (const std::string &status)  
*Creates a RESP simple string (status) response.*
- static std::string [createError](#) (const std::string &error)  
*Creates a RESP error response.*

#### 3.3.1 Detailed Description

Parser for Redis RESP-2 protocol.

Handles serialization and deserialization of Redis RESP-2 protocol messages.

#### 3.3.2 Member Function Documentation

##### 3.3.2.1 createError()

```
std::string RespParser::createError (
    const std::string & error) [static]
```

Creates a RESP error response.

Creates a RESP-2 error response.

##### Parameters

|              |                   |
|--------------|-------------------|
| <i>error</i> | The error message |
|--------------|-------------------|

##### Returns

RESP formatted error message

Error messages start with '-' and contain an error description.

##### Parameters

|              |                    |
|--------------|--------------------|
| <i>error</i> | The error message. |
|--------------|--------------------|

##### Returns

A RESP-2 formatted error response.

### 3.3.2.2 createSimpleString()

```
std::string RespParser::createSimpleString (  
    const std::string & status) [static]
```

Creates a RESP simple string (*status*) response.

Creates a RESP-2 simple string response.

**Parameters**

|               |                    |
|---------------|--------------------|
| <i>status</i> | The status message |
|---------------|--------------------|

**Returns**

RESP formatted simple string

Simple strings are used for success messages and start with '+'.

**Parameters**

|               |                                |
|---------------|--------------------------------|
| <i>status</i> | The success message to return. |
|---------------|--------------------------------|

**Returns**

A RESP-2 formatted simple string.

**3.3.2.3 parseArray()**

```
std::vector< std::string > RespParser::parseArray (
    const std::string & buffer) [static]
```

Parses a RESP array message into command arguments.

Parses a RESP-2 array message and extracts command arguments.

**Parameters**

|               |                         |
|---------------|-------------------------|
| <i>buffer</i> | The RESP message buffer |
|---------------|-------------------------|

**Returns**

Vector of command arguments

RESP-2 arrays start with '\*' followed by the number of elements, and each element is a bulk string prefixed with '\$' followed by its length.

**Parameters**

|               |   |
|---------------|---|
| <i>buffer</i> | The RESP-2 formatted string received from the client. |
|---------------|---|

**Returns**

A vector containing the parsed arguments as strings.

**3.3.2.4 serializeBulkString()**

```
std::string RespParser::serializeBulkString (
    const std::string & value) [static]
```

Serializes a string into RESP bulk string format.

Serializes a string into a RESP-2 bulk string format.



## Parameters

|              |                         |
|--------------|-------------------------|
| <i>value</i> | The string to serialize |
|--------------|-------------------------|

## Returns

RESP formatted bulk string

If the string is empty, it returns the RESP-2 null bulk string "\$-1\r\n".

## Parameters

|              |                              |
|--------------|------------------------------|
| <i>value</i> | The string to be serialized. |
|--------------|------------------------------|

## Returns

A RESP-2 formatted bulk string.

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

- [server/resp\\_parser.h](#)
- [server/resp\\_parser.cpp](#)



# Chapter 4

## File Documentation

### 4.1 benchmarkdata/benchmarkdata.cpp File Reference

Implementation of benchmark data generator.

```
#include "benchmarkdata.h"  
#include <iostream>  
#include <random>  
#include <algorithm>
```

#### 4.1.1 Detailed Description

Implementation of benchmark data generator.

### 4.2 benchmarkdata/benchmarkdata.h File Reference

Benchmark data generator for LSM Tree testing.

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

#### Classes

- class [BenchmarkData](#)  
*Generates and stores test data for benchmarking.*

#### 4.2.1 Detailed Description

Benchmark data generator for LSM Tree testing.

## 4.3 benchmarkdata.h

[Go to the documentation of this file.](#)

```

00001
00005
00006 #ifndef BENCHMARK_DATA_H
00007 #define BENCHMARK_DATA_H
00008
00009 #include <string>
00010 #include <vector>
00011
00019 class BenchmarkData
00020 {
00021 private:
00022     size_t numReads;
00023     size_t numWrites;
00024     size_t keySize;
00025     size_t valueSize;
00026
00032     std::string generateRandomString(size_t length);
00033
00037     void generateTestData();
00038
00039 public:
00040     std::vector<std::string> keys;
00041     std::vector<std::string> values;
00042
00050     BenchmarkData(
00051         size_t reads = 1000000,
00052         size_t writes = 1000000,
00053         size_t keyLength = 16,
00054         size_t valueLength = 16);
00055 };
00056
00057 #endif // BENCHMARK_DATA_H

```

## 4.4 main.cpp File Reference

Main entry point for the BLINK DB server application.

```

#include "server/server.h"
#include "benchmarkdata/benchmarkdata.h"
#include "../part_a/src/StorageEngine/lsmtree.h"
#include <iostream>
#include <string>
#include <stdexcept>

```

### Functions

- `int main()`

### 4.4.1 Detailed Description

Main entry point for the BLINK DB server application.

## 4.5 server/resp\_parser.cpp File Reference

Implementation of Redis RESP-2 protocol parser.

```

#include "resp_parser.h"
#include <sstream>

```

### 4.5.1 Detailed Description

Implementation of Redis RESP-2 protocol parser.

## 4.6 server/resp\_parser.h File Reference

Redis RESP-2 protocol parser implementation header file.

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

### Classes

- class [RespParser](#)  
*Parser for Redis RESP-2 protocol.*

### 4.6.1 Detailed Description

Redis RESP-2 protocol parser implementation header file.

## 4.7 resp\_parser.h

[Go to the documentation of this file.](#)

```
00001
00005
00006 #ifndef RESP_PARSER_H
00007 #define RESP_PARSER_H
00008
00009 #include <string>
00010 #include <vector>
00011
00018 class RespParser
00019 {
00020 public:
00026     static std::vector<std::string> parseArray(const std::string &buffer);
00027
00033     static std::string serializeBulkString(const std::string &value);
00034
00040     static std::string createSimpleString(const std::string &status);
00041
00047     static std::string createError(const std::string &error);
00048 };
00049
00050 #endif // RESP_PARSER_H
```

## 4.8 server/server.cpp File Reference

Implementation of KQueue-based server for handling key-value store operations.

```
#include "server.h"
#include "resp_parser.h"
#include <iostream>
#include <sys/event.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <algorithm>
#include <random>
```

### 4.8.1 Detailed Description

Implementation of KQueue-based server for handling key-value store operations.

## 4.9 server/server.h File Reference

KQueue-based server for LSM Tree storage engine.

```
#include <string>
#include <vector>
#include "../part_a/src/StorageEngine/lsmtree.h"
#include "../benchmarkdata/benchmarkdata.h"
```

### Classes

- class [KQueueServer](#)  
*Server implementation using kqueue for I/O multiplexing.*

### Macros

- #define **PORT** 9002
- #define **MAX\_EVENTS** 1024
- #define **BUFFER\_SIZE** 1024

### 4.9.1 Detailed Description

KQueue-based server for LSM Tree storage engine.

## 4.10 server.h

[Go to the documentation of this file.](#)

```
00001
00005
00006 #ifndef SERVER_H
00007 #define SERVER_H
00008
00009 #include <string>
00010 #include <vector>
00011 #include "../part_a/src/StorageEngine/lsmtree.h"
00012 #include "../benchmarkdata/benchmarkdata.h"
00013
00014 #define PORT 9002
00015 #define MAX_EVENTS 1024
00016 #define BUFFER_SIZE 1024
00017
00025 class KQueueServer
00026 {
00027 private:
00028     int server_fd;
00029     int kq;
00030     LSMTree &store;
00031     BenchmarkData &data;
00032
00039     std::string processCommand(const std::vector<std::string> &args, int rn);
00040
00046     void handleClient(int fd, int rn);
00047
00048 public:
00054     KQueueServer(LSMTree &store, BenchmarkData &data);
00055
00059     ~KQueueServer();
00060
00065     int run();
00066 };
00067
00068 #endif // SERVER_H
```

# Index

- ~KQueueServer
  - KQueueServer, [8](#)
- BenchmarkData, [5](#)
  - BenchmarkData, [6](#)
  - generateRandomString, [6](#)
  - generateTestData, [7](#)
- benchmarkdata/benchmarkdata.cpp, [15](#)
- benchmarkdata/benchmarkdata.h, [15](#), [16](#)
- createError
  - RespParser, [10](#)
- createSimpleString
  - RespParser, [10](#)
- generateRandomString
  - BenchmarkData, [6](#)
- generateTestData
  - BenchmarkData, [7](#)
- handleClient
  - KQueueServer, [8](#)
- KQueueServer, [7](#)
  - ~KQueueServer, [8](#)
  - handleClient, [8](#)
  - KQueueServer, [8](#)
  - processCommand, [9](#)
  - run, [9](#)
- main.cpp, [16](#)
- parseArray
  - RespParser, [12](#)
- processCommand
  - KQueueServer, [9](#)
- RespParser, [10](#)
  - createError, [10](#)
  - createSimpleString, [10](#)
  - parseArray, [12](#)
  - serializeBulkString, [12](#)
- run
  - KQueueServer, [9](#)
- serializeBulkString
  - RespParser, [12](#)
- server/resp\_parser.cpp, [16](#)
- server/resp\_parser.h, [17](#)
- server/server.cpp, [17](#)
- server/server.h, [18](#)