ECE297 Storage Server

0.2

Generated by Doxygen 1.8.1.2

Sat Apr 5 2014 19:02:52

Contents

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 documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 config_params Struct Reference

A struct to store config parameters.

`#include <utils.h>`

Public Attributes

- char server_host [MAX_HOST_LEN]

  The hostname of the server.

- int server_port

  The listening port of the server.

- char username [MAX_USERNAME_LEN]

  The storage server's username.

- char password [MAX_ENC_PASSWORD_LEN]

  The storage server's encrypted password.

- int concurrency

  Multiple Table Names (MAX_TABLES = 100)

- struct string1 **table** [MAX_TABLES]

3.1.1 Detailed Description

A struct to store config parameters.

This struct is for holding server_host, server_port, username, and several table names. These data came fron the configuration file.

Definition at line 80 of file utils.h.

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

- utils.h

3.2 storage_record Struct Reference

Encapsulate the value associated with a key in a table.

`#include <storage.h>`

Public Attributes

- char value [MAX_VALUE_LEN]

  This is where the actual value is stored.

- uintptr_t metadata [8]

  A place to put any extra data.

3.2.1 Detailed Description

Encapsulate the value associated with a key in a table.

The metadata will be used later.

Definition at line 54 of file storage.h.

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

- storage.h

3.3 string Struct Reference

This is the declaration of the struct string, which is used for holding value data inside.

Public Attributes

- char **value** [MAX_VALUE_LEN]
- char **type** [20]
- char **size** [20]
- int **counter**

3.3.1 Detailed Description

This is the declaration of the struct string, which is used for holding value data inside.

Definition at line 118 of file server.c.

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

- server.c

3.4 string1 Struct Reference

Public Attributes

- char **tablename** [MAX_VALUE_LEN]
- struct string2 **column_data** [MAX_COLUMNS_PER_TABLE][3]

3.4.1 Detailed Description

Definition at line 68 of file utils.h.

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

- utils.h

3.5 string2 Struct Reference

A struct to store tablename.

`#include <utils.h>`

Public Attributes

- char **columnvalue** [MAX_COLNAME_LEN]

3.5.1 Detailed Description

A struct to store tablename.

Definition at line 64 of file utils.h.

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

- utils.h

3.6 thread_data Struct Reference

Process a command from the client.

Public Attributes

- int **clientsock**
- struct config_params **params**
- FILE * **file1**

3.6.1 Detailed Description

Process a command from the client.

Parameters

file1	The file stream used in this function.
sock	The socket connected to the client.
cmd	The command received from the client.

Returns

Returns 0 on success, -1 otherwise.

Definition at line 167 of file server.c.

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

- server.c

Chapter 4

File Documentation

4.1 client.c File Reference

This file is used to generate a client shell, which is used to connect to the server part.

```
#include <errno.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "storage.h"
#include <time.h>
#include <sys/time.h>
#include "utils.h"
#include "logger.h"
```

Macros

- #define MAX_CENSUS_CITY_NUM 692

  The maximum city number in census part.

Functions

- void PrintValue (char *the_label)

  Start a client to interact with the storage server.
- int main (int argc, char *argv[])

4.1.1 Detailed Description

This file is used to generate a client shell, which is used to connect to the server part. The client connects to the server, running at SERVERHOST:SERVERPORT and performs a number of storage_* operations. If there are errors, the client exits.

Definition in file client.c.

4.1.2 Function Documentation

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

selection[0] = '\0';

Definition at line 50 of file client.c.

References Authenticated, file, MAX_CENSUS_CITY_NUM, MAX_RECORDS_PER_TABLE, MAX_STRTYPE_SIZE, MAX_VALUE_LEN, storage_record::metadata, storage_auth(), storage_connect(), storage_disconnect(), storage_get(), storage_query(), storage_set(), and storage_record::value.

4.1.2.2 void PrintValue (char * *the_label*)

Start a client to interact with the storage server.

If connect is successful, the client performs a storage_set/get() on TABLE and KEY and outputs the results on stdout. Finally, it exists after disconnecting from the server.

Parameters

argc	The number of strings pointed to by argv
argv[]	An array of pointers to characters

Returns

Return 0 when we want to exit the main function

Definition at line 45 of file client.c.

4.2 encrypt_passwd.c File Reference

This program implements a password encryptor.

```
#include <stdlib.h>
#include <stdio.h>
#include "utils.h"
```

Functions

- void print_usage ()

  Print the usage to stdout.
- int main (int argc, char *argv[])

  This is the main file of the encryption part, which is used to generate the encrypted password.

4.2.1 Detailed Description

This program implements a password encryptor. This file can build a stand-alone program, which can be used to generate encrypted password based on the original password

Definition in file encrypt_passwd.c.

4.2.2 Function Documentation

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

This is the main file of the encryption part, which is used to generate the encrypted password.

Parameters

argc	The number of strings pointed to by argv
argv[]	An array of pointers to characters

Returns

Return 0 if encrypted successfully, otherwise return -1

Definition at line 30 of file encrypt_passwd.c.

References generate_encrypted_password(), and print_usage().

4.3 logger.h File Reference

This file is used to control the output of the log file.

Macros

- #define **LOGGING** 0

4.3.1 Detailed Description

This file is used to control the output of the log file. When LOGGING is 0, the log files will not be outputted. The log files will only be outputted when LOGGING is 2

Definition in file logger.h.

4.4 server.c File Reference

This file implements the storage server.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <string.h>
#include <assert.h>
#include <signal.h>
#include "utils.h"
#include <time.h>
#include <ctype.h>
#include <stdbool.h>
```

Classes

- struct string

  This is the declaration of the struct string, which is used for holding value data inside.
- struct thread_data

  Process a command from the client.

Macros

- #define MAX_CONFIG_LINE_LEN 1024

  Max characters in each config file line.
- #define MAX_USERNAME_LEN 64

  Max characters of server username.
- #define MAX_ENC_PASSWORD_LEN 64

  Max characters of server's encrypted password.
- #define MAX_HOST_LEN 64

  Max characters of server hostname.
- #define MAX_PORT_LEN 8

  Max characters of server port.
- #define MAX_PATH_LEN 256

  Max characters of data directory path.
- #define MAX_TABLES 100

  Max tables supported by the server.
- #define MAX_RECORDS_PER_TABLE 1000

  Max records per table.
- #define MAX_TABLE_LEN 20

  Max characters of a table name.
- #define MAX_KEY_LEN 20

  Max characters of a key name.
- #define MAX_CONNECTIONS 10

  Max simultaneous client connections.

- #define MAX_COLUMNS_PER_TABLE 10

  Max columns per table.
- #define MAX_COLNAME_LEN 20

  Max characters of a column name.
- #define MAX_STRTYPE_SIZE 40

  Max SIZE of string types.
- #define MAX_VALUE_LEN 800

  Max characters of a value.
- #define MAX_LISTENQUEUELEN 20

  The maximum number of queued connections.
- #define **LOGGING** 0
- #define MAX_CENSUS_CITY_NUM 693

  The maximum city numbers.

Functions

- bool TestingCharArray (const char *chararray)

  Check whether a string is actually a valid integer.
- void **ModifyString3** (char *char_source)
- void * **handle_client** (void *data)
- int **handle_command** (FILE *file1, int sock, char *cmd, struct config_params params)
- int main (int argc, char *argv[])

  Start the storage server.

Variables

- struct string TableOfData [MAX_TABLES][MAX_RECORDS_PER_TABLE][11]

  This initialization declares a struct, which is used for holding table of data.
- struct string QueryHolder [MAX_RECORDS_PER_TABLE][MAX_COLUMNS_PER_TABLE+1]

  This initialization declares a struct, which is used for holding query.
- struct string QueryResultHolder [MAX_RECORDS_PER_TABLE][MAX_COLUMNS_PER_TABLE+1]

  This initialization declares a struct, which is used for holding the result of the query.
- int long total_processing_time = 0

  This initialization is for counting the total processing time.
- struct thread_data **thread_data_array** [MAX_CONNECTIONS]

4.4.1 Detailed Description

This file implements the storage server. The storage server should be named "server" and should take a single command line argument that refers to the configuration file.

The storage server should be able to communicate with the client library functions declared in storage.h and implemented in storage.c.

Definition in file server.c.

4.4.2 Function Documentation

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

Start the storage server.

This is the main entry point for the storage server. It reads the configuration file, starts listening on a port, and proccesses commands from clients.

Parameters

argc	The number of strings pointed to by argv
argv[]	An array of pointers to characters

Returns

Return EXIT_SUCCESS if we can exit successfully

Definition at line 1057 of file server.c.

References config_params::concurrency, LOG, logger(), MAX_CMD_LEN, MAX_COLUMNS_PER_TABLE, MAX_CONNECTIONS, MAX_LISTENQUEUELEN, MAX_RECORDS_PER_TABLE, MAX_TABLES, read_config(), recvline(), config_params::server_host, config_params::server_port, and TableOfData.

4.4.2.2 bool TestingCharArray (const char * *chararray*)

Check whether a string is actually a valid integer.

In this function, we handle negative numbers first, then check whether the string is a valid integer or not. If is an integer, return true, otherwise return false

Parameters

chararray	The string to be tested

Returns

Return true if it is integer

Definition at line 67 of file server.c.

4.5 storage.c File Reference

This file contains the implementation of the storage server interface as specified in storage.h.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include "storage.h"
#include "utils.h"
#include "logger.h"
#include <errno.h>
#include <ctype.h>
```

Functions

- void ModifyString (char *char_source)

  This function is used for removing spaces.
- void * storage_connect (const char *hostname, const int port)

  This function is for connecting to the server from client side.
- int storage_auth (const char *username, const char *passwd, void *conn)

  This function is for authentication based on teh username and password inputted by the user.
- int storage_get (const char *table, const char *key, struct storage_record *record, void *conn)

  This function is for users get the value from the storage server.
- int storage_set (const char *table, const char *key, struct storage_record *record, void *conn)

  This function is for users set the value inside the storage server.
- int storage_disconnect (void *conn)

  This function is for disconnecting from the server.
- int storage_query (const char *table, const char *predicates, char **keys, const int max_keys, void *conn)

  This function is for querying databse for keys that fit.

Variables

- int Connected = 0

  This is just a minimal stub implementation. You should modify it according to your design.
- int Authenticated = 0

  Initializing the Authentication for indicating authenticate status, which will be used in the functions below.
- FILE * file

  Initializing a filestream.

4.5.1 Detailed Description

This file contains the implementation of the storage server interface as specified in storage.h. This storage.c file is used by the client side, which includes several different functions. These functions have the following purposes, which include connect, authenticate, get, set and disconnect

Definition in file storage.c.

4.5.2 Function Documentation

4.5.2.1 void ModifyString (char * *char_source*)

This function is used for removing spaces.

When a string be passed inside this function (use pass by parameter, the spaces inside this string will be get ridded.

Parameters

char_source	The string which contain spaces and waiting to be purified

Definition at line 54 of file storage.c.

Referenced by storage_query().

4.5.2.2 int storage_auth (const char * *username,* const char * *passwd,* void * *conn*)

This function is for authentication based on teh username and password inputted by the user.

Authenticate the client's connection to the server.

Parameters

Username	The username inputted by the user Passwd The password inputted by the user This is a void pointer for connection purpose

Returns

Return 0 if successful, otherwise return -1

Definition at line 131 of file storage.c.

References Authenticated, ERR_AUTHENTICATION_FAILED, ERR_INVALID_PARAM, file, generate_encrypted_password(), LOG, logger(), MAX_CMD_LEN, recvline(), and sendall().

Referenced by main().

4.5.2.3 void* storage_connect (const char * *hostname,* const int *port*)

This function is for connecting to the server from client side.

Establish a connection to the server.

Parameters

hostname	The hostname inputted by the user
port	The port inputted by the user

Definition at line 76 of file storage.c.

References Connected, ERR_CONNECTION_FAIL, ERR_INVALID_PARAM, and MAX_PORT_LEN.

Referenced by main().

4.5.2.4 int storage_disconnect (void * *conn*)

This function is for disconnecting from the server.

Close the connection to the server.

Parameters

conn	This is a void pointer for connection purpose

Definition at line 488 of file storage.c.

References Authenticated, ERR_INVALID_PARAM, file, LOG, and logger().

Referenced by main().

4.5.2.5 int storage_get (const char * *table,* const char * *key,* struct storage_record * *record,* void * *conn*)

This function is for users get the value from the storage server.

Retrieve the value associated with a key in a table.

Parameters

table	The name of the table
key	The name of the key
record	Containing the current record
conn	This is a void pointer for connection purpose

Returns

Return 0 if successful, otherwise return -1

Definition at line 198 of file storage.c.

References Authenticated, ERR_INVALID_PARAM, ERR_KEY_NOT_FOUND, ERR_NOT_AUTHENTICATED, ERR_TABLE_NOT_FOUND, ERR_UNKNOWN, file, LOG, logger(), MAX_CMD_LEN, storage_record::metadata, recvline(), sendall(), and storage_record::value.

Referenced by main().

4.5.2.6 int storage_query (const char * *table,* const char * *predicates,* char ** *keys,* const int *max_keys,* void * *conn*)

This function is for querying databse for keys that fit.

Query the table for records, and retrieve the matching keys.

With the help of this function, the program can get the required data from the server, based on the predicates inputted by the user.

Parameters

table	The table of records array
predicates	The conditions given by the user
keys	An array which makes strings as its elements, and used to match the predicates
max_keys	The number of elements inside the array of string "keys"

Returns

If the function be executed successfully, the number of the matching elements is returned. Otherwise, a value of -1 is returned.

Definition at line 536 of file storage.c.

References Authenticated, ERR_INVALID_PARAM, ERR_NOT_AUTHENTICATED, ERR_TABLE_NOT_FOUND, ERR_UNKNOWN, file, LOG, logger(), MAX_CMD_LEN, MAX_STRTYPE_SIZE, ModifyString(), ModifyString2(), recvline(), and sendall().

Referenced by main().

4.5.2.7 int storage_set (const char * *table*, const char * *key*, struct storage_record * *record*, void * *conn*)

This function is for users set the value inside the storage server.

Store a key/value pair in a table.

Parameters

table	The name of the table
key	The name of the key
record	The current record in struct storage_record type
conn	This is a void pointer for connection purpose

Definition at line 348 of file storage.c.

References Authenticated, ERR_INVALID_PARAM, ERR_NOT_AUTHENTICATED, ERR_TABLE_NOT_FOUND, ERR_TRANSACTION_ABORT, ERR_UNKNOWN, file, LOG, logger(), MAX_CMD_LEN, storage_record::metadata, recvline(), sendall(), and storage_record::value.

Referenced by main().

4.5.3 Variable Documentation

4.5.3.1 FILE* file

Initializing a filestream.

Building a filestream named file.

Definition at line 41 of file storage.c.

Referenced by main(), read_config(), storage_auth(), storage_disconnect(), storage_get(), storage_query(), and storage_set().

4.6 storage.h File Reference

This file defines the interface between the storage client and server.

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

Classes

- struct storage_record

  Encapsulate the value associated with a key in a table.

Macros

- #define MAX_CONFIG_LINE_LEN 1024

  Max characters in each config file line.
- #define MAX_USERNAME_LEN 64

  Max characters of server username.
- #define MAX_ENC_PASSWORD_LEN 64

  Max characters of server's encrypted password.
- #define MAX_HOST_LEN 64

  Max characters of server hostname.
- #define MAX_PORT_LEN 8

  Max characters of server port.
- #define MAX_PATH_LEN 256

  Max characters of data directory path.
- #define MAX_TABLES 100

  Max tables supported by the server.
- #define MAX_RECORDS_PER_TABLE 1000

  Max records per table.
- #define MAX_TABLE_LEN 20

  Max characters of a table name.
- #define MAX_KEY_LEN 20

  Max characters of a key name.
- #define MAX_CONNECTIONS 10

  Max simultaneous client connections.
- #define MAX_COLUMNS_PER_TABLE 10

  Max columns per table.
- #define MAX_COLNAME_LEN 20

  Max characters of a column name.
- #define MAX_STRTYPE_SIZE 40

  Max SIZE of string types.
- #define MAX_VALUE_LEN 800

  Max characters of a value.
- #define ERR_INVALID_PARAM 1

  A parameter is not valid.
- #define ERR_CONNECTION_FAIL 2

  Error connecting to server.
- #define ERR_NOT_AUTHENTICATED 3

  Client not authenticated.
- #define ERR_AUTHENTICATION_FAILED 4

  Client authentication failed.
- #define ERR_TABLE_NOT_FOUND 5

  The table does not exist.

- #define ERR_KEY_NOT_FOUND 6

  The key does not exist.
- #define ERR_UNKNOWN 7

  Any other error.
- #define ERR_TRANSACTION_ABORT 8

  Transaction abort error.

Functions

- void * storage_connect (const char *hostname, const int port)

  Establish a connection to the server.
- int storage_auth (const char *username, const char *passwd, void *conn)

  Authenticate the client's connection to the server.
- int storage_get (const char *table, const char *key, struct storage_record *record, void *conn)

  Retrieve the value associated with a key in a table.
- int storage_set (const char *table, const char *key, struct storage_record *record, void *conn)

  Store a key/value pair in a table.
- int storage_query (const char *table, const char *predicates, char **keys, const int max_keys, void *conn)

  Query the table for records, and retrieve the matching keys.
- int storage_disconnect (void *conn)

  Close the connection to the server.

4.6.1 Detailed Description

This file defines the interface between the storage client and server. The functions here should be implemented in storage.c.

You should not modify this file, or else the code used to mark your implementation will break.

Definition in file storage.h.

4.6.2 Function Documentation

4.6.2.1 int storage_auth (const char * *username*, const char * *passwd*, void * *conn*)

Authenticate the client's connection to the server.

Parameters

username	Username to access the storage server.
passwd	Password in its plain text form.
conn	A connection to the server.

Returns

Return 0 if successful, and -1 otherwise.

On error, errno will be set to ERR_AUTHENTICATION_FAILED.

Authenticate the client's connection to the server.

Parameters

Username	The username inputted by the user Passwd The password inputted by the user This is a void pointer for connection purpose

Returns

Return 0 if successful, otherwise return -1

Definition at line 131 of file storage.c.

References Authenticated, ERR_AUTHENTICATION_FAILED, ERR_INVALID_PARAM, file, generate_encrypted_password(), LOG, logger(), MAX_CMD_LEN, recvline(), and sendall().

Referenced by main().

4.6.2.2 void* storage_connect (const char * *hostname*, const int *port*)

Establish a connection to the server.

Parameters

hostname	The IP address or hostname of the server.
port	The TCP port of the server.

Returns

If successful, return a pointer to a data structure that represents a connection to the server. Otherwise return NULL.

On error, errno will be set to one of the following, as appropriate: ERR_INVALID_PARAM, ERR_CONNECTION_FAIL, or ERR_UNKNOWN.

Establish a connection to the server.

Parameters

hostname	The hostname inputted by the user
port	The port inputted by the user

Definition at line 76 of file storage.c.

References Connected, ERR_CONNECTION_FAIL, ERR_INVALID_PARAM, and MAX_PORT_LEN.

Referenced by main().

4.6.2.3 int storage_disconnect (void * *conn*)

Close the connection to the server.

Parameters

conn	A pointer to the connection structure returned in an earlier call to storage_connect().

Returns

Return 0 if successful, and -1 otherwise.

On error, errno will be set to one of the following, as appropriate: ERR_INVALID_PARAM, ERR_CONNECTION_FAIL, or ERR_UNKNOWN.

Close the connection to the server.

Parameters

conn	This is a void pointer for connection purpose

Definition at line 488 of file storage.c.

References Authenticated, ERR_INVALID_PARAM, file, LOG, and logger().

Referenced by main().

4.6.2.4 int storage_get (const char * *table*, const char * *key*, struct storage_record * *record*, void * *conn*)

Retrieve the value associated with a key in a table.

Parameters

table	A table in the database.
key	A key in the table.
record	A pointer to a record struture.
conn	A connection to the server.

Returns

Return 0 if successful, and -1 otherwise.

On error, errno will be set to one of the following, as appropriate: ERR_INVALID_PARAM, ERR_CONNECTION_FAIL, ERR_TABLE_NOT_FOUND, ERR_KEY_NOT_FOUND, ERR_NOT_AUTHENTICATED, or ERR_UNKNOWN.

The record with the specified key in the specified table is retrieved from the server using the specified connection. If the key is found, the record structure is populated with the details of the corresponding record. Otherwise, the record structure is not modified.

Retrieve the value associated with a key in a table.

Parameters

table	The name of the table
key	The name of the key
record	Containing the current record
conn	This is a void pointer for connection purpose

Returns

Return 0 if successful, otherwise return -1

Definition at line 198 of file storage.c.

References Authenticated, ERR_INVALID_PARAM, ERR_KEY_NOT_FOUND, ERR_NOT_AUTHENTICATED, ERR_TABLE_NOT_FOUND, ERR_UNKNOWN, file, LOG, logger(), MAX_CMD_LEN, storage_record::metadata, recvline(), sendall(), and storage_record::value.

Referenced by main().

4.6.2.5 int storage_query (const char * *table,* const char * *predicates,* char ** *keys,* const int *max_keys,* void * *conn*)

Query the table for records, and retrieve the matching keys.

Parameters

table	A table in the database.
predicates	A comma separated list of predicates.
keys	An array of strings where the keys whose records match the specified predicates will be copied. The array must have room for at least max_keys elements. The caller must allocate memory for this array.
max_keys	The size of the keys array.
conn	A connection to the server.

Returns

Return the number of matching keys (which may be more than max_keys) if successful, and -1 otherwise.

On error, errno will be set to one of the following, as appropriate: ERR_INVALID_PARAM, ERR_CONNECTION_FAIL, ERR_TABLE_NOT_FOUND, ERR_KEY_NOT_FOUND, ERR_NOT_AUTHENTICATED, or ERR_UNKNOWN.

Each predicate consists of a column name, an operator, and a value, each separated by optional whitespace. The operator may be a "=" for string types, or one of "$<$, $>$, =" for int and float types. An example of query predicates is "name = bob, mark $>$ 90".

Query the table for records, and retrieve the matching keys.

With the help of this function, the program can get the required data from the server, based on the predicates inputted by the user.

Parameters

table	The table of records array
predicates	The conditions given by the user
keys	An array which makes strings as its elements, and used to match the predicates
max_keys	The number of elements inside the array of string "keys"

Returns

If the function be executed successfully, the number of the matching elements is returned. Otherwise, a value of -1 is returned.

Definition at line 536 of file storage.c.

References Authenticated, ERR_INVALID_PARAM, ERR_NOT_AUTHENTICATED, ERR_TABLE_NOT_FOUND, ERR_UNKNOWN, file, LOG, logger(), MAX_CMD_LEN, MAX_STRTYPE_SIZE, ModifyString(), ModifyString2(), recvline(), and sendall().

Referenced by main().

4.6.2.6 int storage_set (const char * *table,* const char * *key,* struct **storage_record** * *record,* void * *conn*)

Store a key/value pair in a table.

Parameters

table	A table in the database.
key	A key in the table.
record	A pointer to a record struture.
conn	A connection to the server.

Returns

Return 0 if successful, and -1 otherwise.

On error, errno will be set to one of the following, as appropriate: ERR_INVALID_PARAM, ERR_CONNECTION_FAIL, ERR_TABLE_NOT_FOUND, ERR_KEY_NOT_FOUND, ERR_NOT_AUTHENTICATED, or ERR_UNKNOWN.

The key and record are stored in the table of the database using the connection. If the key already exists in the table, the corresponding record is updated with the one specified here. If the key exists in the table and the record is NULL, the key/value pair are deleted from the table.

Store a key/value pair in a table.

Parameters

table	The name of the table
key	The name of the key
record	The current record in struct storage_record type
conn	This is a void pointer for connection purpose

Definition at line 348 of file storage.c.

References Authenticated, ERR_INVALID_PARAM, ERR_NOT_AUTHENTICATED, ERR_TABLE_NOT_FOUND, ERR_TRANSACTION_ABORT, ERR_UNKNOWN, file, LOG, logger(), MAX_CMD_LEN, storage_record::metadata, recvline(), sendall(), and storage_record::value.

Referenced by main().

4.7 utils.c File Reference

This file implements various utility functions that are can be used by the storage server and client library.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdbool.h>
#include "utils.h"
```

Functions

- void ModifyString2 (char *char_source)

  This function is used for removing spaces.
- int TestingStringCorrectness (char *char_array_tested)

This function is used for testing the validity of the line.

- int sendall (const int sock, const char *buf, const size_t len)

  This function is used for sending data through the socket.
- int recvline (const int sock, char *buf, const size_t buflen)

  Receive an entire line from a socket.
- int process_config_line (char *line, struct config_params *params)

  Parse and process a line in the config file.
- int read_config (const char *config_file, struct config_params *params)

  Read and load configuration parameters.
- void logger (FILE *file, char *message)

  Generates a log message.
- char * generate_encrypted_password (const char *passwd, const char *salt)

  Generates an encrypted password string using salt CRYPT_SALT.

4.7.1 Detailed Description

This file implements various utility functions that are can be used by the storage server and client library. Most of the functions in this file have different utilities, while can be used by server side or client side in order to connect with each other, send data, read configuration file, process configuration file and generate encrypted password for security purpose

Definition in file utils.c.

4.7.2 Function Documentation

4.7.2.1 char* generate_encrypted_password (const char * *passwd*, const char * *salt*)

Generates an encrypted password string using salt CRYPT_SALT.

Parameters

passwd	Password before encryption.
salt	Salt used to encrypt the password. If NULL default value DEFAULT_CRYPT_SALT is used.

Returns

Returns encrypted password.

Definition at line 627 of file utils.c.

References DEFAULT_CRYPT_SALT.

Referenced by main(), and storage_auth().

4.7.2.2 void logger (FILE * *file*, char * *message*)

Generates a log message.

Parameters

file	The output stream
message	Message to log.

Definition at line 611 of file utils.c.

Referenced by main(), storage_auth(), storage_disconnect(), storage_get(), storage_query(), and storage_set().

4.7.2.3 void ModifyString2 (char * *char_source*)

This function is used for removing spaces.

When a string be passed inside this function (pass by parameter), the spaces inside this string will be get removed.

Parameters

char_source	The string which contain spaces about to be removed

Definition at line 34 of file utils.c.

Referenced by process_config_line(), and storage_query().

4.7.2.4 int process_config_line (char * *line,* struct **config_params** * *params*)

Parse and process a line in the config file.

Parameters

line	Containing a line of strings
params	Containing the data taken from configuration file

char ColumnNameArray[MAX_COLNAME_LEN*MAX_COLUMNS_PER_TABLE]; ///For storing each column name inside char ColumnTypeArray[MAX_COLNAME_LEN*MAX_COLUMNS_PER_TABLE]; ///For storing each column type inside

printf("column2: %s \n\n", token2);

Definition at line 141 of file utils.c.

References config_params::concurrency, MAX_COLNAME_LEN, MAX_COLUMNS_PER_TABLE, MAX_CONFIG_LINE_LEN, MAX_STRTYPE_SIZE, MAX_TABLES, ModifyString2(), config_params::password, config_params::server_host, config_params::server_port, TestingStringCorrectness(), and config_params::username.

Referenced by read_config().

4.7.2.5 int read_config (const char * *config_file,* struct **config_params** * *params*)

Read and load configuration parameters.

Parameters

config_file	The name of the configuration file.
params	The structure where config parameters are loaded.

Returns

Return 0 on success, -1 otherwise.

Definition at line 558 of file utils.c.

References file, MAX_CONFIG_LINE_LEN, config_params::password, process_config_line(), config_params::server_host, config_params::server_port, and config_params::username.

Referenced by main().

4.7.2.6 int recvline (const int *sock*, char * *buf*, const size_t *buflen*)

Receive an entire line from a socket.

Returns

Return 0 on success, -1 otherwise

Parameters

sock	The socket which will be used
buf	A buffer for storing string inside
buflen	The length of the buffer

Definition at line 106 of file utils.c.

Referenced by main(), storage_auth(), storage_get(), storage_query(), and storage_set().

4.7.2.7 int sendall (const int *sock*, const char * *buf*, const size_t *len*)

This function is used for sending data through the socket.

Keep sending the contents of the buffer until complete.

Parameters

sock	The socket number
buf	The buf for characters
len	The length of the characters with type const size_t

Definition at line 80 of file utils.c.

Referenced by storage_auth(), storage_get(), storage_query(), and storage_set().

4.7.2.8 int TestingStringCorrectness (char * *char_array_tested*)

This function is used for testing the validity of the line.

This function is used for testing whether a string has a right format

Parameters

char_array_tested	The string to be tested

Definition at line 57 of file utils.c.

Referenced by process_config_line().

4.8 utils.h File Reference

This file declares various utility functions that are can be used by the storage server and client library.

```
#include <stdio.h>
#include "storage.h"
#include <stdbool.h>
```

Classes

- struct string2

  A struct to store tablename.
- struct string1
- struct config_params

  A struct to store config parameters.

Macros

- #define MAX_CMD_LEN (1024 * 8)

  The max length in bytes of a command from the client to the server.
- #define LOG(x) {printf x; fflush(stdout);}

  A macro to log some information.
- #define DBG(x) {printf x; fflush(stdout);}

  A macro to output debug information.
- #define DEFAULT_CRYPT_SALT "xx"

  Default two character salt used for password encryption.

Functions

- int sendall (const int sock, const char *buf, const size_t len)

  Keep sending the contents of the buffer until complete.
- int recvline (const int sock, char *buf, const size_t buflen)

  Receive an entire line from a socket.
- int read_config (const char *config_file, struct config_params *params)

  Read and load configuration parameters.
- void logger (FILE *file, char *message)

  Generates a log message.
- char * generate_encrypted_password (const char *passwd, const char *salt)

  Generates an encrypted password string using salt CRYPT_SALT.

Variables

- FILE * file

  Building a filestream named file.

4.8.1 Detailed Description

This file declares various utility functions that are can be used by the storage server and client library. In this header file, we have some macros for limiting correnponding maximum numbers. Besides, there is also a structure declaration called config_params in this file. The config_params structure is for holding configuration data. There are also some function declarations, which will be used in utils.c

Definition in file utils.h.

4.8.2 Macro Definition Documentation

4.8.2.1 #define DBG(*x*) {printf x; fflush(stdout);}

A macro to output debug information.

It is only enabled in debug builds.

Definition at line 57 of file utils.h.

4.8.2.2 #define LOG(*x*) {printf x; fflush(stdout);}

A macro to log some information.

Use it like this: LOG(("Hello %s", "world\n"))

Don't forget the double parentheses, or you'll get weird errors!

Definition at line 47 of file utils.h.

Referenced by main(), storage_auth(), storage_disconnect(), storage_get(), storage_query(), and storage_set().

4.8.3 Function Documentation

4.8.3.1 char* generate_encrypted_password (const char * *passwd,* const char * *salt*)

Generates an encrypted password string using salt CRYPT_SALT.

Parameters

passwd	Password before encryption.
salt	Salt used to encrypt the password. If NULL default value DEFAULT_CRYPT_SALT is used.

Returns

Returns encrypted password.

Definition at line 627 of file utils.c.

References DEFAULT_CRYPT_SALT.

Referenced by main(), and storage_auth().

4.8.3.2 void logger (FILE * *file,* char * *message*)

Generates a log message.

Parameters

file	The output stream
message	Message to log.

Definition at line 611 of file utils.c.

Referenced by main(), storage_auth(), storage_disconnect(), storage_get(), storage_query(), and storage_set().

4.8.3.3 int read_config (const char * *config_file*, struct config_params * *params*)

Read and load configuration parameters.

Parameters

config_file	The name of the configuration file.
params	The structure where config parameters are loaded.

Returns

Return 0 on success, -1 otherwise.

Definition at line 558 of file utils.c.

References file, MAX_CONFIG_LINE_LEN, config_params::password, process_config_line(), config_params::server_host, config_params::server_port, and config_params::username.

Referenced by main().

4.8.3.4 int recvline (const int *sock*, char * *buf*, const size_t *buflen*)

Receive an entire line from a socket.

Returns

Return 0 on success, -1 otherwise.
Return 0 on success, -1 otherwise

Parameters

sock	The socket which will be used
buf	A buffer for storing string inside
buflen	The length of the buffer

Definition at line 106 of file utils.c.

Referenced by main(), storage_auth(), storage_get(), storage_query(), and storage_set().

4.8.3.5 int sendall (const int *sock*, const char * *buf*, const size_t *len*)

Keep sending the contents of the buffer until complete.

Returns

Return 0 on success, -1 otherwise.

The parameters mimic the send() function.

Keep sending the contents of the buffer until complete.

Parameters

sock	The socket number
buf	The buf for characters
len	The length of the characters with type const size_t

Definition at line 80 of file utils.c.

Referenced by storage_auth(), storage_get(), storage_query(), and storage_set().

Index