

Lab 3 - CE156 - Reliable UDP”

Generated by Doxygen 1.8.1.2

Sat May 10 2014 13:48:09

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

client_ip_port	??
mftp_packet		
My custom protocol packet	??
threadargs	??

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

client.c	??
rudp.c	??
rudp.h	??
server.c	??
utils.c	??
utils.h	??

Chapter 3

Data Structure Documentation

3.1 client_ip_port Struct Reference

Data Fields

- unsigned long [ip](#)
- unsigned short [port](#)

3.1.1 Field Documentation

3.1.1.1 unsigned long client_ip_port::ip

3.1.1.2 unsigned short client_ip_port::port

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

- [server.c](#)

3.2 mftp_packet Struct Reference

My custom protocol packet.

```
#include <rudp.h>
```

Data Fields

- unsigned int [seq](#)
- char [data](#) [1024]
- unsigned int [flag](#)

3.2.1 Detailed Description

My custom protocol packet.

3.2.2 Field Documentation

3.2.2.1 `char mftp_packet::data[1024]`

3.2.2.2 `unsigned int mftp_packet::flag`

3.2.2.3 `unsigned int mftp_packet::seq`

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

- [rudp.h](#)

3.3 threadargs Struct Reference

Data Fields

- unsigned int [cnum](#)
- char [address](#) [128]
- unsigned int [port](#)
- char [filename](#) [256]
- unsigned int [validipnum](#)

3.3.1 Field Documentation

3.3.1.1 `char threadargs::address[128]`

3.3.1.2 `unsigned int threadargs::cnum`

3.3.1.3 `char threadargs::filename[256]`

3.3.1.4 `unsigned int threadargs::port`

3.3.1.5 `unsigned int threadargs::validipnum`

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

- [client.c](#)

Chapter 4

File Documentation

4.1 client.c File Reference

```
#include <arpa/inet.h>
#include <errno.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include "utils.h"
#include "rudp.h"
```

Data Structures

- struct [threadargs](#)

Macros

- #define [SUCCESS](#) 0
- #define [FAILURE](#) 1
- #define [FALSE](#) 0

Typedefs

- typedef struct [sockaddr](#) sockaddr
- typedef struct [sockaddr_in](#) sockaddr_in

Functions

- void [check_error](#) (char *x, char *y)
- void * [thread_get_chunk](#) (void *arg)
- unsigned char * [serialize_threadargs](#) (struct [threadargs](#), unsigned char buffer[])

- struct [threadargs deserialize_threadargs](#) (unsigned char buffer[])
- int [main](#) (int argc, char **argv)

4.1.1 Macro Definition Documentation

4.1.1.1 `#define FAILURE 1`

4.1.1.2 `#define FALSE 0`

4.1.1.3 `#define SUCCESS 0`

4.1.2 Typedef Documentation

4.1.2.1 `typedef struct sockaddr sockaddr`

4.1.2.2 `typedef struct sockaddr_in sockaddr_in`

4.1.3 Function Documentation

4.1.3.1 `void check_error (char * x, char * y)`

4.1.3.2 `struct threadargs deserialize_threadargs (unsigned char buffer[])` [read]

4.1.3.3 `int main (int argc, char ** argv)`

4.1.3.4 `unsigned char * serialize_threadargs (struct threadargs p, unsigned char buffer[])`

4.1.3.5 `void * thread_get_chunk (void * arg)`

4.2 rudp.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdarg.h>
#include <dirent.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include "rudp.h"
#include "utils.h"
```

Macros

- `#define SUCCESS 0`
- `#define FAILURE 1`
- `#define TRUE 1`
- `#define FALSE 0`

Functions

- unsigned char * [serialize_int](#) (unsigned char *buffer, unsigned int val)
Serializes an int into a unsigned char.
- unsigned char * [serialize_data](#) (unsigned char *buffer, char buf[], int len)
Serializes an char array into a unsigned char array.
- unsigned char * [serialize_packet](#) ([mftp_packet](#) packet, unsigned char buffer[])
Serialize [mftp_packet](#) into a buffer.
- unsigned char * [deserialize_int](#) (unsigned char *buffer, unsigned int *val)
Deserializes an int into a unsigned char.
- unsigned char * [deserialize_data](#) (unsigned char *buffer, char buf[], int len)
Deserializes an char array into a unsigned char array.
- [mftp_packet](#) [deserialize_packet](#) (unsigned char buffer[])
Deserialize buffer into [mftp_packet](#).
- void [send_error](#) (int seq, int clisock, const struct [sockaddr_in](#) client, int clen)
Send an error datagram to a socket.
- void [send_ack](#) (int seq, int clisock, const struct [sockaddr_in](#) client, int clen)
Send an ack datagram to a socket.
- int [send_dgram](#) (int socket, const struct [sockaddr_in](#) *cli, socklen_t dlen, const [mftp_packet](#) data)
Sends a datagram to a client.
- [mftp_packet](#) [parse_dgram](#) (unsigned char buffer[])
Parses incoming datagrams.

4.2.1 Macro Definition Documentation

4.2.1.1 `#define FAILURE 1`

4.2.1.2 `#define FALSE 0`

4.2.1.3 `#define SUCCESS 0`

4.2.1.4 `#define TRUE 1`

4.2.2 Function Documentation

4.2.2.1 unsigned char* [deserialize_data](#) (unsigned char * *buffer*, char *buf*[], int *len*)

Deserializes an char array into a unsigned char array.

Parameters

<i>buffer</i>	The array to get the data from.
<i>buf</i>	The buffer to save it.
<i>len</i>	Then length of buf.

Returns

A pointer to the next free space in the buffer.

4.2.2.2 unsigned char* [deserialize_int](#) (unsigned char * *buffer*, unsigned int * *val*)

Deserializes an int into a unsigned char.

Parameters

<i>buffer</i>	The array to get the data out of.
<i>val</i>	The value to save the data.

Returns

A pointer to the next free space in the buffer.

4.2.2.3 `mftp_packet deserialize_packet (unsigned char buffer[])`

Deserialize buffer into `mftp_packet`.

Parameters

<i>buffer</i>	The buffer to get data from.
<i>len</i>	The length of the buffer.

4.2.2.4 `mftp_packet parse_dgram (unsigned char buffer[])`

Parses incoming datagrams.

runs the deserializer.

Parameters

<i>buffer</i>	The buffer to parse into a mftp struct
---------------	--

Returns

A `mftp_packet` filled with data.

4.2.2.5 `void send_ack (int sequence_number, int clisock, struct sockaddr_in client, int clen)`

Send an ack datagram to a socket.

Parameters

<i>sequence_number</i>	The sequence number of the datagram
<i>clisock</i>	The socket to send the data to.
<i>client</i>	The reciever.
<i>clen</i>	The length of the struct sockaddr_in.

4.2.2.6 `int send_dgram (int socket, const struct sockaddr_in * cli, socklen_t dlen, const mftp_packet data)`

Sends a datagram to a client.

Parameters

<i>socket</i>	The socket to send to.
<i>cli</i>	the structure with the ip and port to send to.
<i>dlen</i>	length of the stucture cli.
<i>data</i>	The custom packet with data and flags etc.

Returns

Returns 1 if successful and 0 if it fails. Appropriate messages are printed to stderr.

4.2.2.7 void send_error (int *sequence_number*, int *clisock*, sockaddr_in *client*, int *clen*)

Send an error datagram to a socket.

Parameters

<i>sequence_number</i>	The sequence number of the datagram
<i>clisock</i>	The socket to send the data to.
<i>client</i>	The reciever.
<i>clen</i>	The length of the sockaddr_in struct.

4.2.2.8 unsigned char* serialize_data (unsigned char * *buffer*, char *buf*[], int *len*)

Serializes an char array into a unsigned char array.

Parameters

<i>buffer</i>	The array to insert the data.
<i>buf</i>	The value to serialize.
<i>len</i>	Then length of buf.

Returns

A pointer to the next free space in the buffer.

4.2.2.9 unsigned char* serialize_int (unsigned char * *buffer*, unsigned int *val*)

Serializes an int into a unsigned char.

Parameters

<i>buffer</i>	The array to insert the data.
<i>val</i>	The value to serialize.

Returns

A pointer to the next free space in the buffer.

4.2.2.10 unsigned char* serialize_packet (mftp_packet *packet*, unsigned char *buffer*[])

Serialize [mftp_packet](#) into a buffer.

Parameters

<i>packet</i>	The packet to be serialized into a buffer.
<i>buffer</i>	The buffer to fill up/
<i>len</i>	The length of the buffer.

4.3 rudp.h File Reference

Data Structures

- struct [mftp_packet](#)
My custom protocol packet.

Macros

- #define [START](#) 1
Flags for custom packet.
- #define [DATA](#) 2
- #define [ACK](#) 3
- #define [ERROR](#) 4

Typedefs

- typedef struct [sockaddr](#) [sockaddr](#)
- typedef struct [sockaddr_in](#) [sockaddr_in](#)
- typedef struct [mftp_packet](#) [mftp_packet](#)
My custom protocol packet.
- typedef [mftp_packet](#) * [mftp_packet_ref](#)

Functions

- unsigned char * [serialize_int](#) (unsigned char *buffer, unsigned int val)
Serializes an int into a unsigned char.
- unsigned char * [serialize_data](#) (unsigned char *buffer, char buf[], int len)
Serializes an char array into a unsigned char array.
- unsigned char * [deserialize_int](#) (unsigned char *buffer, unsigned int *val)
Deserializes an int into a unsigned char.
- unsigned char * [deserialize_data](#) (unsigned char *buffer, char buf[], int len)
Deserializes an char array into a unsigned char array.
- unsigned char * [serialize_packet](#) ([mftp_packet](#) packet, unsigned char buffer[])
Serialize [mftp_packet](#) into a buffer.
- [mftp_packet](#) [deserialize_packet](#) (unsigned char buffer[])
Deserialize buffer into [mftp_packet](#).
- void [send_ack](#) (int sequence_number, int clisock, [sockaddr_in](#) client, int clen)
Send an ack datagram to a socket.
- void [send_error](#) (int sequence_number, int clisock, [sockaddr_in](#) client, int clen)
Send an error datagram to a socket.
- int [send_dgram](#) (int socket, const struct [sockaddr_in](#) *cli, socklen_t dlen, const [mftp_packet](#) data)
Sends a datagram to a client.
- [mftp_packet](#) [parse_dgram](#) (unsigned char buffer[])
Parses incoming datagrams.

4.3.1 Macro Definition Documentation

4.3.1.1 `#define ACK 3`

4.3.1.2 `#define DATA 2`

4.3.1.3 `#define ERROR 4`

4.3.1.4 `#define START 1`

Flags for custom packet.

4.3.2 Typedef Documentation

4.3.2.1 `typedef struct mftp_packet mftp_packet`

My custom protocol packet.

4.3.2.2 `typedef mftp_packet* mftp_packet_ref`

4.3.2.3 `typedef struct sockaddr sockaddr`

4.3.2.4 `typedef struct sockaddr_in sockaddr_in`

4.3.3 Function Documentation

4.3.3.1 `unsigned char* deserialize_data (unsigned char * buffer, char buf[], int len)`

Deserializes an char array into a unsigned char array.

Parameters

<i>buffer</i>	The array to get the data from.
<i>buf</i>	The buffer to save it.
<i>len</i>	Then length of buf.

Returns

A pointer to the next free space in the buffer.

4.3.3.2 `unsigned char* deserialize_int (unsigned char * buffer, unsigned int * val)`

Deserializes an int into a unsigned char.

Parameters

<i>buffer</i>	The array to get the data out of.
<i>val</i>	The value to save the data.

Returns

A pointer to the next free space in the buffer.

4.3.3.3 mftp_packet deserialize_packet (unsigned char *buffer*[])

Deserialize buffer into [mftp_packet](#).

Parameters

<i>buffer</i>	The buffer to get data from.
<i>len</i>	The length of the buffer.

4.3.3.4 mftp_packet parse_dgram (unsigned char *buffer*[])

Parses incoming datagrams.

runs the deserializer.

Parameters

<i>buffer</i>	The buffer to parse into a mftp struct
---------------	--

Returns

A [mftp_packet](#) filled with data.

4.3.3.5 void send_ack (int *sequence_number*, int *clisock*, struct sockaddr_in *client*, int *clen*)

Send an ack datagram to a socket.

Parameters

<i>sequence_number</i>	The sequence number of the datagram
<i>clisock</i>	The socket to send the data to.
<i>client</i>	The reciever.
<i>clen</i>	The length of the sockaddr_in struct.

4.3.3.6 int send_dgram (int *socket*, const struct sockaddr_in * *cli*, socklen_t *dlen*, const mftp_packet *data*)

Sends a datagram to a client.

Parameters

<i>socket</i>	The socket to send to.
<i>cli</i>	the structure with the ip and port to send to.
<i>dlen</i>	length of the stucture cli.
<i>data</i>	The custom packet with data and flags etc.

Returns

Returns 1 if successful and 0 if it fails. Appropriate messages are printed to stderr.

4.3.3.7 void send_error (int *sequence_number*, int *clisock*, struct sockaddr_in *client*, int *clen*)

Send an error datagram to a socket.

Parameters

<i>sequence_ - number</i>	The sequence number of the datagram
<i>clisock</i>	The socket to send the data to.
<i>client</i>	The reciever.
<i>crlen</i>	The length of the sockaddr_in struct.

4.3.3.8 unsigned char* serialize_data (unsigned char * *buffer*, char *buf*[], int *len*)

Serializes an char array into a unsigned char array.

Parameters

<i>buffer</i>	The array to insert the data.
<i>buf</i>	The value to serialize.
<i>len</i>	Then length of buf.

Returns

A pointer to the next free space in the buffer.

4.3.3.9 unsigned char* serialize_int (unsigned char * *buffer*, unsigned int *val*)

Serializes an int into a unsigned char.

Parameters

<i>buffer</i>	The array to insert the data.
<i>val</i>	The value to serialize.

Returns

A pointer to the next free space in the buffer.

4.3.3.10 unsigned char* serialize_packet (mftp_packet *packet*, unsigned char *buffer*[])

Serialize [mftp_packet](#) into a buffer.

Parameters

<i>packet</i>	The packet to be serialized into a buffer.
<i>buffer</i>	The buffer to fill up/
<i>len</i>	The length of the buffer.

4.4 server.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <errno.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/select.h>
#include <time.h>
#include <sys/ioctl.h>
#include <pthread.h>
#include "utils.h"
#include "rudp.h"
```

Data Structures

- struct [client_ip_port](#)

Macros

- #define [SUCCESS](#) 0
- #define [FAILURE](#) 1

Typedefs

- typedef struct [sockaddr](#) [sockaddr](#)
- typedef struct [sockaddr_in](#) [sockaddr_in](#)

Functions

- void * [handle_client_request](#) (void *clisock)
- void [close_client](#) (int clisock, fd_set *master)
- unsigned char * [serialize_datastruct](#) (struct [client_ip_port](#) p, unsigned char buffer[])
- struct [client_ip_port](#) [deserialize_datastruct](#) (unsigned char buffer[])
- int [main](#) (int argc, char **argv)

4.4.1 Macro Definition Documentation

4.4.1.1 #define [FAILURE](#) 1

4.4.1.2 #define [SUCCESS](#) 0

4.4.2 Typedef Documentation

4.4.2.1 typedef struct [sockaddr](#) [sockaddr](#)

4.4.2.2 typedef struct sockaddr_in sockaddr_in

4.4.3 Function Documentation

4.4.3.1 void close_client (int *clisock*, fd_set * *master*)

4.4.3.2 struct client_ip_port deserialize_datastruct (unsigned char *buffer[]*) [read]

4.4.3.3 void * handle_client_request (void * *clisock*)

4.4.3.4 int main (int *argc*, char ** *argv*)

4.4.3.5 unsigned char * serialize_datastruct (struct client_ip_port *p*, unsigned char *buffer[]*)

4.5 utils.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdarg.h>
#include <dirent.h>
#include <sys/stat.h>
#include "utils.h"
#include "rudp.h"
```

Macros

- #define SUCCESS 0
- #define FAILURE 1

Functions

- void [check_socket](#) (int fd)
Checks the return value of the socket(3) networking api call for any errors and prints messages and sets the exit status accordingly.
- void [check_connection](#) (int x)
Checks the return value of the connect(3) networking api call for any errors and prints messages and sets the exit status accordingly.
- FILE * [retrieve_file](#) (const char *restrict filename, const char *restrict mode)
Checks the current directory for the file filename.
- int [get_file_size](#) (FILE *restrict file)
Takes a file and gives back the size of the file.
- [mftp_packet_get_file_chunk](#) (int f_offset, FILE *restrict stream, int seq)
Gets a chunk of a file to a client socket.
- void [debugprintf](#) (char *format,...)

4.5.1 Macro Definition Documentation

4.5.1.1 #define FAILURE 1

4.5.1.2 #define SUCCESS 0

4.5.2 Function Documentation

4.5.2.1 void check_connection (int *val*)

Checks the return value of the connect(3) networking api call for any errors and prints messages and sets the exit status accordingly.

Parameters

<i>val</i>	The return value from the connect(3) call.
------------	--

4.5.2.2 void check_socket (int *fd*)

Checks the return value of the socket(3) networking api call for any errors and prints messages and sets the exit status accordingly.

Parameters

<i>fd</i>	The file descriptor returned by the socket(3) call.
-----------	---

4.5.2.3 void debugprintf (char * *format*, ...)

4.5.2.4 mftp_packet_get_file_chunk (int *f_offset*, FILE **restrict stream*, int *seq*)

Gets a chunk of a file to a client socket.

Parameters

<i>f_offset</i>	The offset to index into the file.
<i>stream</i>	The file to read from to send the chunk.
<i>seq</i>	The sequence number.

4.5.2.5 int get_file_size (FILE **restrict filename*)

Takes a file and gives back the size of the file.

Parameters

<i>filename</i>	The file in which you want the size of.
-----------------	---

Returns

The size of the file filename, or -1 if an error occurs. Errno will be set to the proper error.

4.5.2.6 FILE* retrieve_file (const char **restrict filename*, const char **restrict mode*)

Checks the current directory for the file filename.

If file name is found retrieve_file will attempt to open the file using fopen. If not an error message will be returned.

Parameters

<i>filename</i>	The file to be searched for and opened.
<i>mode</i>	The mode in which the file will be opened.

Returns

The file descriptor for the file if fopen succeeds. Otherwise NULL is returned if filename is not found, or if fopen fails.

4.6 utils.h File Reference

```
#include "rudp.h"
```

Macros

- `#define DEBUGF(...) debugprintf (__VA_ARGS__)`
Allows for debugging print statements to be made and easily turned off for release build.

Enumerations

- enum `bool` { `FALSE` = 0, `TRUE` = 1 }
A boolean data type created by an enum.

Functions

- void `check_socket` (int fd)
Checks the return value of the socket(3) networking api call for any errors and prints messages and sets the exit status accordingly.
- void `check_connection` (int val)
Checks the return value of the connect(3) networking api call for any errors and prints messages and sets the exit status accordingly.
- FILE * `retrieve_file` (const char *restrict filename, const char *restrict mode)
Checks the current directory for the file filename.
- int `get_file_size` (FILE *restrict filename)
Takes a file and gives back the size of the file.
- `mftp_packet get_file_chunk` (int f_offset, FILE *restrict stream, int seq)
Gets a chunk of a file to a client socket.
- void `debugprintf` (char *format,...)

4.6.1 Macro Definition Documentation

4.6.1.1 #define DEBUGF(...) debugprintf (__VA_ARGS__)

Allows for debugging print statements to be made and easily turned off for release build.

Parameters

<i>format</i>	The format string to format the print statement.
---------------	--

4.6.2 Enumeration Type Documentation

4.6.2.1 enum bool

A boolean data type created by an enum.

Enumerator:

FALSE

TRUE

4.6.3 Function Documentation

4.6.3.1 void check_connection (int val)

Checks the return value of the connect(3) networking api call for any errors and prints messages and sets the exit status accordingly.

Parameters

<i>val</i>	The return value from the connect(3) call.
------------	--

4.6.3.2 void check_socket (int fd)

Checks the return value of the socket(3) networking api call for any errors and prints messages and sets the exit status accordingly.

Parameters

<i>fd</i>	The file descriptor returned by the socket(3) call.
-----------	---

4.6.3.3 void debugprintf (char * format, ...)

4.6.3.4 mftp_packet get_file_chunk (int f_offset, FILE *restrict stream, int seq)

Gets a chunk of a file to a client socket.

Parameters

<i>f_offset</i>	The offset to index into the file.
<i>stream</i>	The file to read from to send the chunk.
<i>seq</i>	The sequence number.

4.6.3.5 int get_file_size (FILE *restrict filename)

Takes a file and gives back the size of the file.

Parameters

<i>filename</i>	The file in which you want the size of.
-----------------	---

Returns

The size of the file filename, or -1 if an error occurs. Errno will be set to the proper error.

4.6.3.6 FILE* retrieve_file (const char *restrict filename, const char *restrict mode)

Checks the current directory for the file filename.

If file name is found retrieve_file will attempt to open the file using fopen. If not an error message will be returned.

Parameters

<i>filename</i>	The file to be searched for and opened.
<i>mode</i>	The mode in which the file will be opened.

Returns

The file descriptor for the file if fopen succeeds. Otherwise NULL is returned if filename is not found, or if fopen fails.