qmesg

0.1

Generated by Doxygen 1.15.0

Chapter 1

Bug List

File irc.c

None known

File main.c

Possible race conditions during authentication resulting in misses.

Unexpected cuts in responses from the server

Seems to get stuck in a livelock somewhere resulting in not being able to input commands

File response.c

None known

File session.c

None known

File user.c

None known

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 _session_t Struct Reference

Public Attributes

- struct sockaddr_in **adr**
- int **fd**
- int **status**

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

- include/client_session.h

4.2 _user_t Struct Reference

Public Attributes

- char ∗ **password**
- char ∗ **nickname**
- char ∗ **username**
- char ∗ **realname**

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

- include/user.h

4.3 client_message_t Struct Reference

Public Attributes

- char ∗ **name**
- int **len**
- int(∗ **func**)(int fd, char ∗args)

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

- include/command.h

Chapter 5

File Documentation

5.1 client_session.h

```
#ifndef CLIENT_SESSION_H
#define CLIENT_SESSION_H
// Q: Should these be included here?
// Q: Can I remove some of these?
#include <arpa/inet.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <unistd.h>

#define MAX_ADDR_LEN 100
#define DOMAIN AF_INET // IPv4
#define TYPE SOCK_STREAM // TCP
#define PROTOCOL 0 // use provided protocol configuration

typedef struct _session_t {
    struct sockaddr_in adr; // Q: Should this be a pointer?
    int fd;
    int status; // 0: open connection, 1: closed connection
} session_t;

session_t *session_init(session_t *srv, int prt, char * adr, int adr_len);
extern int session_destroy(session_t *srv);
extern int session_connect(session_t *srv);
extern int session_disconnect(session_t *srv);
extern int session_fd(session_t *sesh);
// TODO: Make save and load function
#endif

```

5.2 command.h

```
#ifndef COMMAND_H
#define COMMAND_H

typedef struct client_message_t {
    char * name;
    int len;
    int (*func)(int fd, char * args);
} client_message;

extern int parse(int fd, char *msg, int len);
#endif
```

5.3 common.h

```
#ifndef COMMON_H
#define COMMON_H
```

```c

#define MSG_MAX_LEN 1024
#define PASSWORD_MAX_LEN 30
#define NICKNAME_MAX_LEN 30
#define USERNAME_MAX_LEN 30
#define REALNAME_MAX_LEN 30
#define IP_MAX_LEN 16
#define PORT_MAX_LEN 5
#define CMD_MAX_LEN 8
#endif
```

5.4 irc.h

```c
#include <string.h>
#include <unistd.h>
#include <common.h> // Provides common magic literals

extern int pass(int fd,  char *password);
extern int nick(int fd,  char *nickname);
extern int user(int fd,  char *username, char *realname);
extern int ping(int fd, char *token);
extern int pong(int fd, char *token) ;
extern int join(int fd, char *channel);
extern int quit(int fd);
extern int privmsg(int fd, char *channel, char *msg);
extern int list(int fd);
extern int topic(int fd, char *channel);
extern int part(int fd, char *channel);
```

5.5 user.h

```c
#ifndef USER_H
#define USER_H
#define PASSWORD_MAX_LEN 30
#define NICKNAME_MAX_LEN 30
#define USERNAME_MAX_LEN 30
#define REALNAME_MAX_LEN 30

typedef struct _user_t {
    char *password;
    char *nickname;
    char *username;
    char *realname;
} user_t;

extern user_t *user_init(user_t *usr, char *psw, char *nck, char *usn, char *rln);
extern void user_destroy(user_t *usr);
// TODO: Make save and load function

// Setters for user values
extern int set_password(user_t *usr, char *psw);
extern int set_nickname(user_t *usr, char *nck);
extern int set_username(user_t *usr, char *usn);
extern int set_realname(user_t *usr, char *rln);

#endif

```

5.6 src/linux/main.c File Reference

Main.

```c
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <signal.h>
#include <stdbool.h>
#include <netinet/in.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <poll.h>
#include <fcntl.h>
#include <user.h>
#include <irc.h>
#include <client_session.h>
#include <command.h>
#include <common.h>
```

Macros

- #define **TITLE** "qmesg - linux client\n"
- #define **INTERRUPT_MSG** "\nCaught interrupt\n"
- #define **NUM_FEEDS** 2
- #define **TEST_PASSWORD** "psw"
- #define **TEST_NICKNAME** "nck"
- #define **TEST_USERNAME** "usr"
- #define **TEST_REALNAME** "rln"
- #define **TEST_PORT** 6667
- #define **TEST_ADDR** "127.0.0.1"

Functions

- int **main** (int argc, char const ∗argv[ ])

5.6.1 Detailed Description

Main.

Author

Aqiel Oostenbrug

Date

December 3, 2025

Version

1.1

Bug Possible race conditions during authentication resulting in misses.

Unexpected cuts in responses from the server

Seems to get stuck in a livelock somewhere resulting in not being able to input commands

5.7 src/linux/response.c File Reference

Provide responses for messages received from a server.

```
#include <string.h>
#include <command.h>
#include <common.h>
#include <stdio.h>
#include <irc.h>
```

Functions

- int parse (int fd, char *response, int response_len)

  Execute command corresponding to `response` with `response_len`.

Variables

- client_message cm_table [ ]

5.7.1 Detailed Description

Provide responses for messages received from a server.

Author

Aqiel Oostenbrug

Date

December 3, 2025

Version

1.1

Bug None known

5.7.2 Function Documentation

5.7.2.1 parse()

```
int parse (
            int fd,
            char * response,
            int response_len)
```

Execute command corresponding to `response` with `response_len`.

Parameters

fd	file descriptor for the command to write to
response	response to parse
response_len	length of the response to pars

Returns

cm_table[i].func(fd, params) if successful
0 if unmatched,
-1 if `fd` is invalid,
-2 if response is invalid,
-3 if response is too short,
-4 if response only includes a client message parameter,
-5 if response only includes a space as a client message parameter

5.7.3 Variable Documentation

5.7.3.1 cm_table

```
client_message cm_table[]
```

Initial value:

```
= {
    {"PING", 4, respond_pong}
}
```

5.8 src/linux/session.c File Reference

Manage TCP client-side sessions.

```
#include <client_session.h>
#include <stdlib.h>
#include <stdio.h>
#include <netinet/tcp.h>
```

Functions

- session_t ∗ session_init (session_t ∗sesh, int prt, char ∗adr, int adr_len)

  Initialize `sesh` with `prt`, `adr`, and `adr_len`.
- int session_destroy (session_t ∗sesh)

  Destroy `sesh`.
- int session_connect (session_t ∗sesh)

  Connect to the server specified by `sesh`.
- int session_disconnect (session_t ∗sesh)

  Disconnect from the server specified by `sesh`.
- int session_fd (session_t ∗sesh)

  Return the file descriptor granted by `sesh`.

5.8.1 Detailed Description

Manage TCP client-side sessions.

Author

Aqiel Oostenbrug

Date

December 3, 2025

Version

1.0

Bug None known

5.8.2 Function Documentation

5.8.2.1 session_connect()

```
int session_connect (
            session_t * sesh)
```

Connect to the server specified by `sesh`.

Parameters

sesh	session to start

Returns

0 if successful,
1 otherwise

5.8.2.2 session_destroy()

```
int session_destroy (
            session_t * sesh)
```

Destroy `sesh`.

Parameters

sesh	session to destroy

Returns

0 if successful
1 if `sesh` was not initialized

5.8.2.3 session_disconnect()

```
int session_disconnect (
            session_t * sesh)
```

Disconnect from the server specified by `sesh`.

Parameters

sesh	session to end

Returns

0 if successful,
1 otherwise

5.8.2.4 session_fd()

```
int session_fd (
            session_t * sesh)
```

Return the file descriptor granted by `sesh`.

Parameters

sesh	to source the file descriptor from

Returns

`sesh->fd` file descriptor of `sesh`
-1 otherwise

5.8.2.5 session_init()

```
session_t * session_init (
            session_t * sesh,
            int prt,
            char * adr,
            int adr_len)
```

Initialize `sesh` with `prt`, `adr`, and `adr_len`.

Parameters

sesh	session to initialize
prt	port to connect to
adr	ip address to connect to
adr_len	length of `adr`

Returns

sesh if successful,
NULL otherwise

5.9 src/shared/irc.c File Reference

Provide IRC commands for Client to Server commnication.

```
#include <string.h>
#include <unistd.h>
#include <common.h>
#include <irc.h>
```

Functions

- int pass (int fd, char *password)

  write `password` to the server.
- int nick (int fd, char *nickname)

  write `nickname` to the server.
- int user (int fd, char *username, char *realname)

  write `username` and `realname` to the server.
- int ping (int fd, char *token)

  Ping the server.
- int pong (int fd, char *token)

  Pong the server.
- int join (int fd, char *channel)

  Join `server`.
- int quit (int fd)

  Terminate the connection to the server.
- int privmsg (int fd, char *channel, char *msg)

  write `msg` `channel` to the server.
- int list (int fd)

  Request the list of channels.
- int topic (int fd, char *channel)

  Request the topic of `channel`.
- int part (int fd, char *channel)

  Request to leave `channel`.

5.9.1 Detailed Description

Provide IRC commands for Client to Server commnication.

Note that since write is used no fflush is required!

Author

Aqiel Oostenbrug

Date

November 26, 2025

Version

1.1

Bug None known

See also

`https://modern.ircdocs.horse/`

5.9.2 Function Documentation

5.9.2.1 join()

```
int join (
            int fd,
            char * channel)
```

Join `server`.

Parameters

in	*fd*	file descriptor to write to
in	*channel*	channel to join

Returns

0 if successful,
1 if `fd` is invalid,
3 if `channel` is invalid,
4 if message is too long

5.9.2.2 list()

```
int list (
            int fd)
```

Request the list of channels.

Parameters

`in`	*fd*	file descriptor to write to

Returns

0 if request send successfully,
1 if `fd` is invalid

5.9.2.3 nick()

```
int nick (
            int fd,
            char * nickname)
```

write `nickname` to the server.

Parameters

`in`	*fd*	file descriptor to write to
`in`	*nickname*	nickname to write

Returns

0 if successful,
1 if `fd` is invalid,
2 if `nickname` is invalid,
3 if message is too long

5.9.2.4 part()

```
int part (
            int fd,
            char * channel)
```

Request to leave `channel`.

Parameters

`in`	*fd*	file descriptor to write to
`in`	*channel*	channel to leave

Returns

0 if successful,
1 if `fd` is invalid,
2 if `channel` is invalid,
3 if total message is too long

5.9.2.5 pass()

```
int pass (
            int fd,
            char * password)
```

write `password` to the server.

Parameters

in	*fd*	file descriptor to write to
in	*password*	password to write

Returns

0 if successful,
1 if `fd` is invalid,
2 if `password` is invalid,
3 if message is too long

5.9.2.6 ping()

```
int ping (
            int fd,
            char * token)
```

Ping the server.

Parameters

in	*fd*	file descriptor to write to
in	*token*	token to write

Returns

0 if successful,
1 if `fd` is invalid,
3 if `token` is invalid,
4 if message is too long

5.9.2.7 pong()

```
int pong (
            int fd,
            char * token)
```

Pong the server.

Parameters

in	*fd*	file descriptor to write to
in	*token*	token to write

Returns

0 if successful,
1 if `fd` is invalid,
3 if `token` is invalid,
4 if message is too long

5.9.2.8 privmsg()

```
int privmsg (
            int fd,
            char * channel,
            char * msg)
```

write `msg` `channel` to the server.

Parameters

	fd	file descriptor to write to
`in`	*channel*	channel to write
`in`	*msg*	message to write

Returns

0 if request send successfully,
1 if `fd` is invalid,
2 if `channel` is invalid,
3 if `msg` is invalid,
4 if total message is too long

5.9.2.9 quit()

```
int quit (
            int fd)
```

Terminate the connection to the server.

Parameters

`in`	*fd*	file descriptor to write to
`in`	*token*	token to write

Returns

0 if successful,
1 if `fd` is invalid

5.9.2.10 topic()

```
int topic (
            int fd,
            char * channel)
```

Request the topic of `channel`.

Parameters

in	*fd*	file descriptor to write to
in	*channel*	channel from which the topic to request

Returns

0 if successful,
1 if `fd` is invalid,
2 if `channel` is invalid,
3 if total message is too long

5.9.2.11 user()

```
int user (
            int fd,
            char * username,
            char * realname)
```

write `username` and `realname` to the server.

Parameters

	fd	file descriptor to write to
in	*username*	username to write
in	*realname*	realname to write

Returns

0 if successful,
1 if `fd` is invalid,
2 if `username` is invalid,
3 if `realname` is invalid,
4 if message is too long

5.10 src/shared/user.c File Reference

Manage user data.

```
#include <user.h>
#include <string.h>
#include <stdlib.h>
```

Functions

- user_t ∗ user_init (user_t ∗usr, char ∗psw, char ∗nck, char ∗usn, char ∗rln)

  Initialize `usr` *with* `psw`*,* `prt`*,* `adr`*, and* `adr_len`*.*
- void user_destroy (user_t ∗usr)

  Destroy `usr`*.*
- int set_password (user_t ∗usr, char ∗psw)

  Set `psw` *to* `usr`*.*
- int set_nickname (user_t ∗usr, char ∗nck)

  Set `nck` *to* `usr`*.*
- int set_username (user_t ∗usr, char ∗usn)

  Set `usn` *to* `usr`*.*
- int set_realname (user_t ∗usr, char ∗rln)

  Set `rln` *to* `usr`*.*

5.10.1 Detailed Description

Manage user data.

Author

Aqiel Oostenbrug

Date

December 3, 2025

Version

1.0

Bug None known

5.10.2 Function Documentation

5.10.2.1 set_nickname()

```
int set_nickname (
            user_t * usr,
            char * nck)
```

Set `nck` to `usr`.

Parameters

usr	user to update

nck	nickname to set

Returns

0 if successful, 1 otherwise

5.10.2.2 set_password()

```
int set_password (
            user_t * usr,
            char * psw)
```

Set `psw` to `usr`.

Parameters

usr	user to update
psw	password to set

Returns

0 if successful, 1 otherwise

5.10.2.3 set_realname()

```
int set_realname (
            user_t * usr,
            char * rln)
```

Set `rln` to `usr`.

Parameters

usr	user to update
rln	realname to set

Returns

0 if successful, 1 otherwise

5.10.2.4 set_username()

```
int set_username (
            user_t * usr,
            char * usn)
```

Set `usn` to `usr`.

Parameters

usr	user to update
usn	username to set

Returns

0 if successful, 1 otherwise

5.10.2.5 user_destroy()

```
void user_destroy (
            user_t * usr)
```

Destroy `usr`.

Parameters

usr	user to destroy

5.10.2.6 user_init()

```
user_t * user_init (
            user_t * usr,
            char * psw,
            char * nck,
            char * usn,
            char * rln)
```

Initialize `usr` with `psw`, `prt`, `adr`, and `adr_len`.

Parameters

usr	`user` to initialize
psw	`password` of `user`
nck	`nickname` of `user`
usn	`username` of `user`
rln	`realname` of `user`

Returns

`usr` if successful, `NULL` otherwise

Index