

Sublim Telegram

1.00a

Generated by Doxygen 1.8.7

Thu Mar 17 2016 16:10:05

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	dictionary Struct Reference	5
3.1.1	Field Documentation	5
3.1.1.1	sz	5
3.1.1.2	words	5
3.2	msgToRoomStruct Struct Reference	5
3.2.1	Field Documentation	5
3.2.1.1	msg	5
3.2.1.2	roomName	5
3.3	rooms Struct Reference	6
3.3.1	Field Documentation	6
3.3.1.1	room	6
3.3.1.2	sz	6
3.4	users Struct Reference	6
3.4.1	Field Documentation	6
3.4.1.1	name	6
3.4.1.2	socks	6
3.4.1.3	sz	6
4	File Documentation	7
4.1	.dep.inc File Reference	7
4.2	client.c File Reference	7
4.2.1	Function Documentation	8
4.2.1.1	envoi_message	8
4.2.1.2	main	8
4.2.1.3	reception_message	8

4.2.1.4	stop	8
4.2.2	Variable Documentation	9
4.2.2.1	adresse_locale	9
4.2.2.2	host	9
4.2.2.3	pseudo	9
4.2.2.4	pseudoWithComa	9
4.2.2.5	pseudoWithComaAndRoom	9
4.2.2.6	ptr_host	9
4.2.2.7	ptr_service	9
4.2.2.8	salon	9
4.2.2.9	socket_descriptor	9
4.2.2.10	t1	9
4.2.2.11	t2	9
4.3	client.h File Reference	9
4.3.1	Typedef Documentation	10
4.3.1.1	hostent	10
4.3.1.2	servent	10
4.3.1.3	sockaddr	10
4.3.1.4	sockaddr_in	10
4.3.2	Function Documentation	10
4.3.2.1	envoi_message	10
4.3.2.2	main	10
4.3.2.3	reception_message	10
4.3.2.4	stop	11
4.4	main.c File Reference	11
4.4.1	Function Documentation	11
4.4.1.1	main	11
4.5	server.c File Reference	11
4.5.1	Function Documentation	12
4.5.1.1	addUser	12
4.5.1.2	addUserInRoom	12
4.5.1.3	afficherRooms	13
4.5.1.4	analyseMessage	13
4.5.1.5	findRoomFromSocket	13
4.5.1.6	getServerResponse	13
4.5.1.7	incrementInsult	14
4.5.1.8	main	14
4.5.1.9	readWords	14
4.5.1.10	removeRoom	14
4.5.1.11	removeSocketFromRoom	15

4.5.1.12	renvoi_message	15
4.5.1.13	sendMessageToRoom	15
4.5.1.14	stop	15
4.5.1.15	updateUserFile	16
4.5.2	Variable Documentation	16
4.5.2.1	currentSock	16
4.5.2.2	dict	16
4.5.2.3	ips	16
4.5.2.4	mutexRoom	16
4.5.2.5	mutexUserFile	16
4.5.2.6	room	16
4.5.2.7	t1	16
4.6	server.h File Reference	16
4.6.1	Macro Definition Documentation	18
4.6.1.1	TAILLE_MAX_NOM	18
4.6.2	Typedef Documentation	18
4.6.2.1	hostent	18
4.6.2.2	servent	18
4.6.2.3	sockaddr	18
4.6.2.4	sockaddr_in	18
4.6.3	Function Documentation	18
4.6.3.1	addUser	18
4.6.3.2	addUserInRoom	18
4.6.3.3	afficherRooms	18
4.6.3.4	analyseMessage	19
4.6.3.5	findRoomFromSocket	20
4.6.3.6	getServerResponse	20
4.6.3.7	incrementInsult	20
4.6.3.8	main	20
4.6.3.9	readWords	21
4.6.3.10	removeRoom	21
4.6.3.11	removeSocketFromRoom	21
4.6.3.12	renvoi_message	22
4.6.3.13	sendMessageToRoom	22
4.6.3.14	stop	22
4.6.3.15	updateUserFile	22
	Index	24

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

dictionary	5
msgToRoomStruct	5
rooms	6
users	6

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

.dep.inc	7
client.c	7
client.h	9
main.c	11
server.c	11
server.h	16

Chapter 3

Data Structure Documentation

3.1 dictionary Struct Reference

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

Data Fields

- int [sz](#)
- char [words](#) [50][19]

3.1.1 Field Documentation

3.1.1.1 int [sz](#)

3.1.1.2 char [words](#)[50][19]

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

- [server.h](#)

3.2 msgToRoomStruct Struct Reference

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

Data Fields

- char * [roomName](#)
- char * [msg](#)

3.2.1 Field Documentation

3.2.1.1 char* [msg](#)

3.2.1.2 char* [roomName](#)

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

- [server.h](#)

3.3 rooms Struct Reference

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

Data Fields

- int [sz](#)
- [users room](#) [10]

3.3.1 Field Documentation

3.3.1.1 [users room](#)[10]

3.3.1.2 [int sz](#)

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

- [server.h](#)

3.4 users Struct Reference

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

Data Fields

- char * [name](#)
- int [sz](#)
- int [socks](#) [10]

3.4.1 Field Documentation

3.4.1.1 [char* name](#)

3.4.1.2 [int socks](#)[10]

3.4.1.3 [int sz](#)

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

- [server.h](#)

Chapter 4

File Documentation

4.1 .dep.inc File Reference

4.2 client.c File Reference

```
#include "client.h"
```

Functions

- void [stop](#) ()
Fonction d'arrêt du client, ferme le socket et les threads.
- void * [envoi_message](#) (void *arg)
Fonction threadée d'envoi de message, se ferme une fois le message envoyé
- void * [reception_message](#) (void *arg)
Boucle de reception, traitement et affichage du message. Thread unique contenant une boucle infinie. s'arrête a la deconnexion du client.
- int [main](#) (int argc, char **argv)
fonction main, se connecte au serveur et recupere les entrées client

Variables

- [sockaddr_in](#) [adresse_locale](#)
- [hostent](#) * [ptr_host](#)
- [servent](#) * [ptr_service](#)
- [pthread_t](#) [t1](#)
- [pthread_t](#) [t2](#)
- int [socket_descriptor](#)
- char * [host](#)
- char * [pseudo](#)
- char * [salon](#)
- char [pseudoWithComa](#) [14]
- char [pseudoWithComaAndRoom](#) [26]

4.2.1 Function Documentation

4.2.1.1 void* envoi_message (void * *arg*)

Fonction threadée d'envoi de message, se ferme une fois le message envoyé

envoi_message

Parameters

<i>arg,message</i>	ecrit par le client
--------------------	---------------------

Returns

void

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

fonction main, se connecte au serveur et recupere les entrées client

main, ouvre le serveur et se met en attente de connexions

main

Parameters

<i>argc</i>	
<i>argv</i>	

Returns

1 en cas d'echec, 0 sinon

4.2.1.3 void* reception_message (void * *arg*)

Boucle de reception, traitement et affichage du message. Thread unique contenant une boucle infinie. s'arrête a la deconnexion du client.

reception_message

Parameters

<i>arg</i>	le message reçu par le client
------------	-------------------------------

Returns

void

4.2.1.4 void stop ()

Fonction d'arrêt du client, ferme le socket et les threads.

Ferme le server après avoir fermé toutes les room.

[stop\(\)](#)

Returns

void

4.2.2 Variable Documentation

4.2.2.1 sockaddr_in adresse_locale

4.2.2.2 char* host

4.2.2.3 char* pseudo

4.2.2.4 char pseudoWithComa[14]

4.2.2.5 char pseudoWithComaAndRoom[26]

4.2.2.6 hostent* ptr_host

4.2.2.7 servent* ptr_service

4.2.2.8 char* salon

4.2.2.9 int socket_descriptor

4.2.2.10 pthread_t t1

4.2.2.11 pthread_t t2

4.3 client.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <linux/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <pthread.h>
#include <signal.h>
```

Typedefs

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

Functions

- void [stop](#) ()
Fonction d'arrêt du client, ferme le socket et les threads.
- void * [envoi_message](#) (void *arg)
Fonction threadée d'envoi de message, se ferme une fois le message envoyé
- void * [reception_message](#) (void *arg)
Boucle de reception, traitement et affichage du message. Thread unique contenant une boucle infinie. s'arrête a la deconnexion du client.
- int [main](#) (int argc, char **argv)
fonction main, se connecte au serveur et recupere les entrées client

4.3.1 Typedef Documentation

4.3.1.1 typedef struct hostent hostent

4.3.1.2 typedef struct servent servent

4.3.1.3 typedef struct sockaddr sockaddr

4.3.1.4 typedef struct sockaddr_in sockaddr_in

4.3.2 Function Documentation

4.3.2.1 void* envoi_message (void * *arg*)

Fonction threadée d'envoi de message, se ferme une fois le message envoyé

envoi_message

Parameters

<i>arg,message</i>	ecrit par le client
--------------------	---------------------

Returns

void

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

fonction main, se connecte au serveur et recupere les entrées client

main

Parameters

<i>argc</i>	
<i>argv</i>	

Returns

1 en cas d'echec, 0 sinon

4.3.2.3 void* reception_message (void * *arg*)

Boucle de reception, traitement et affichage du message. Thread unique contenant une boucle infinie. s'arrête a la deconnexion du client.

reception_message

Parameters

<i>arg</i>	le message reçu par le client
------------	-------------------------------

Returns

void

4.3.2.4 void stop ()

Fonction d'arrêt du client, ferme le socket et les threads.

`stop()`

Returns

void

4.4 main.c File Reference

Functions

- int `main` (int argc, char **argv)
fonction main, se connecte au serveur et recupere les entrées client

4.4.1 Function Documentation

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

fonction main, se connecte au serveur et recupere les entrées client

main, ouvre le serveur et se met en attente de connexions

main

Parameters

<i>argc</i>	
<i>argv</i>	

Returns

1 en cas d'echec, 0 sinon

4.5 server.c File Reference

```
#include "server.h"
```

Functions

- char * `findRoomFromSocket` (int sock)
Parcourt chaque room à la recherche du socket, et renvoie la room qui correspond.
- int `removeRoom` (char *roomName)
Retire la room désirée, et décale toutes les autres dans la structure.
- int `removeSocketFromRoom` (int sock, char *roomName)
Supprime le socket d'une room et décale tous les autres.
- void `afficherRooms` ()
affiche toutes les rooms et leur socket, coté serveur.
- void * `sendMessageToRoom` (void *rMsg)
Fonction threadee d'envoi de message a tous les sockets d'une room.
- void `readWords` (dictionary *d)

- Ouvre le fichier de mots et lit chaque ligne pour les stocker dans le dictionnaire.*
- int `updateUserFile` (`sockaddr_in` adresse)
vérifie la présence d'un utilisateur dans le fichier client
- int `incrementInsult` (`sockaddr_in` adresse)
incrémente le nombre d'insultes d'un client, et le kick si supérieur à 15 a chaque message
- int `addUser` (`users` *u, int *sock)
Ajoute un utilisateur dans la room en parametre.
- int `addUserInRoom` (int *sock, char *roomName)
Crée la room, ou la retrouve, puis appelle addUser pour ajouter l'utilisateur dedans.
- char * `analyseMessage` (char *message, `dictionnary` *d, int *sock)
analyse le message, et incrémente le nombre d'insulte s'il y en a une
- char * `getServerResponse` (char *commandLine)
analyse la commande du client pour agir en fonction
- void * `renvoi_message` (void *arg)
Boucle de reception et envoi du message a la room.
- void `stop` ()
Fonction d'arrêt du client, ferme le socket et les threads.
- `main` (int argc, char **argv)
fonction main, se connecte au serveur et recupere les entrées client

Variables

- `rooms` room
- `dictionnary` dict
- int `currentSock`
- pthread_mutex_t `mutexUserFile`
- pthread_mutex_t `mutexRoom`
- `sockaddr_in` ips [104]
- pthread_t t1

4.5.1 Function Documentation

4.5.1.1 int addUser (users * u, int * sock)

Ajoute un utilisateur dans la room en parametre.

addUser

Parameters

<i>u</i>	le salon
<i>sock</i>	le socket

Returns

1 si ajout reussi, 0 sinon

4.5.1.2 int addUserInRoom (int * sock, char * roomName)

Crée la room, ou la retrouve, puis appelle addUser pour ajouter l'utilisateur dedans.

addUserInRoom

Parameters

<i>sock</i>	le socket
<i>roomName</i>	le nom de la room

Returns

1 si ajout reussi, 0 sinon

4.5.1.3 void afficherRooms ()

affiche toutes les rooms et leur socket, coté serveur.

afficherRooms

Returns

void

4.5.1.4 char* analyseMessage (char * *message*, dictionnary * *d*, int * *sock*)

analyse le message, et incrémente le nombre d'insulte s'il y en a une

analyseMMessage

Parameters

<i>message</i>	le message a analyser
<i>d</i>	le dictionnaire de mots
<i>sock</i>	le socket du message envoyé

Returns

le message

4.5.1.5 char* findRoomFromSocket (int *sock*)

Parcourt chaque room à la recherche du socket, et renvoie la room qui correspond.

findRoomFromSocket

Parameters

<i>sock</i>	le socket a chercher
-------------	----------------------

Returns

char* room, la room où se trouve le socket

4.5.1.6 char* getServerResponse (char * *commandLine*)

analyse la commande du client pour agir en fonction

getServerResponse

Parameters

<i>commandLine</i>	la commande utilisateur entrée par le client
--------------------	--

Returns

la reponse du serveur qui correspond

4.5.1.7 int incrementInsult (sockaddr_in adresse)

incremente le nombre d'insultes d'un client, et le kick si supérieur à 15 a chaque message

incrementInsult

Parameters

<i>adresse</i>	l'ip du client
----------------	----------------

Returns

1 si l'incrementation est effectuée, -1 sinon

4.5.1.8 main (int argc, char ** argv)

fonction main, se connecte au serveur et recupere les entrées client

main, ouvre le serveur et se met en attente de connexions

main

Parameters

<i>argc</i>	
<i>argv</i>	

Returns

1 en cas d'echec, 0 sinon

4.5.1.9 void readWords (dictionary * d)

Ouvre le fichier de mots et lit chaque ligne pour les stocker dans le dictionnaire.

readWords

Parameters

<i>d</i>	le dictionnaire a remplir
----------	---------------------------

Returns

void

4.5.1.10 int removeRoom (char * roomName)

Retire la room désirée, et décale toutes les autres dans la structure.

removeRoom

Parameters

<i>roomName</i>	le nom de la room a supprimer
-----------------	-------------------------------

Returns

1 si la suppression est reussie, 0 sinon

4.5.1.11 int removeSocketFromRoom (int *sock*, char * *roomName*)

Supprime le socket d'une room et decalle tous les autres.

removeSocketFromRoom

Parameters

<i>sock</i>	le socket a supprimer
<i>roomName</i>	la room ou se trouve le socket

Returns

1 si suppression reussie, 0 sinon.

4.5.1.12 void* renvoi_message (void * *arg*)

Boucle de reception et envoi du message a la room.

renvoi_message

Parameters

<i>arg</i>	le message envoyé par le client
------------	---------------------------------

Returns

void

4.5.1.13 void* sendMessageToRoom (void * *rMsg*)

Fonction threadee d'envoi de message a tous les sockets d'une room.

sendMessageToRoom

Parameters

<i>rMsg</i>	le message a envoyer
-------------	----------------------

Returns

void

4.5.1.14 void stop ()

Fonction d'arrêt du client, ferme le socket et les threads.

Ferme le server après avoir fermé toutes les room.

[stop\(\)](#)

Returns

void

4.5.1.15 int updateUserFile (sockaddr_in adresse)

vérifie la présence d'un utilisateur dans le fichier client

updateUserFile

Parameters

<i>adresse</i>	l'ip du client
----------------	----------------

Returns

retourne 1 si l'utilisateur est nouveau, 0 si déjà connu, -1 si erreur

4.5.2 Variable Documentation**4.5.2.1 int currentSock****4.5.2.2 dictionary dict****4.5.2.3 sockaddr_in ips[104]****4.5.2.4 pthread_mutex_t mutexRoom****4.5.2.5 pthread_mutex_t mutexUserFile****4.5.2.6 rooms room****4.5.2.7 pthread_t t1****4.6 server.h File Reference**

```
#include <stdlib.h>
#include <stdio.h>
#include <linux/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <pthread.h>
#include <signal.h>
#include <unistd.h>
#include <time.h>
```

Data Structures

- struct [users](#)
- struct [rooms](#)
- struct [dictionary](#)
- struct [msgToRoomStruct](#)

Macros

- #define [TAILLE_MAX_NOM](#) 256

Typedefs

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

Functions

- char * [findRoomFromSocket](#) (int sock)
Parcourt chaque room à la recherche du socket, et renvoie la room qui correspond.
- int [removeRoom](#) (char *roomName)
Retire la room désirée, et décale toutes les autres dans la structure.
- int [removeSocketFromRoom](#) (int sock, char *roomName)
Supprime le socket d'une room et decalle tous les autres.
- void [readWords](#) (dictionary *d)
Ouvre le fichier de mots et lit chaque ligne pour les stocker dans le dictionnaire.
- int [incrementInsult](#) ([sockaddr_in](#) adresse)
incrémente le nombre d'insultes d'un client, et le kick si supérieur à 15 a chaque message
- int [updateUserFile](#) ([sockaddr_in](#) adresse)
vérifie la présence d'un utilisateur dans le fichier client
- int [addUser](#) (users *u, int *sock)
Ajoute un utilisateur dans la room en parametre.
- int [addUserInRoom](#) (int *sock, char *roomName)
Crée la room, ou la retrouve, puis appelle addUser pour ajouter l'utilisateur dedans.
- void [afficherRooms](#) ()
affiche toutes les rooms et leur socket, coté serveur.
- char * [analyseMessage](#) (char *message, dictionary *d, int *sock)
analyse le message, et incrémente le nombre d'insulte s'il y en a une
- char * [getServerResponse](#) (char *commandLine)
analyse la commande du client pour agir en fonction
- void * [renvoi_message](#) (void *arg)
Boucle de reception et envoi du message a la room.
- void * [sendMessageToRoom](#) (void *rMsg)
Fonction threadée d'envoi de message a tous les sockets d'une room.
- void [stop](#) ()
Ferme le server après avoir fermé toutes les room.
- [main](#) (int argc, char **argv)
main, ouvre le serveur et se met en attente de connexions

4.6.1 Macro Definition Documentation

4.6.1.1 `#define TAILLE_MAX_NOM 256`

4.6.2 Typedef Documentation

4.6.2.1 `typedef struct hostent hostent`

4.6.2.2 `typedef struct servent servent`

4.6.2.3 `typedef struct sockaddr sockaddr`

4.6.2.4 `typedef struct sockaddr_in sockaddr_in`

4.6.3 Function Documentation

4.6.3.1 `int addUser (users * u, int * sock)`

Ajoute un utilisateur dans la room en parametre.

`addUser`

Parameters

<i>u</i>	le salon
<i>sock</i>	le socket

Returns

1 si ajout reussi, 0 sinon

4.6.3.2 `int addUserInRoom (int * sock, char * roomName)`

Crée la room, ou la retrouve, puis appelle `addUser` pour ajouter l'utilisateur dedans.

`addUserInRoom`

Parameters

<i>sock</i>	le socket
<i>roomName</i>	le nom de la room

Returns

1 si ajout reussi, 0 sinon

4.6.3.3 `void afficherRooms ()`

affiche toutes les rooms et leur socket, coté serveur.

`afficherRooms`

Returns

`void`

4.6.3.4 `char* analyseMessage (char * message, dictionnary * d, int * sock)`

analyse le message, et incrémente le nombre d'insulte s'il y en a une
analyseMessage

Parameters

<i>message</i>	le message a analyser
<i>d</i>	le dictionnaire de mots
<i>sock</i>	le socket du message envoyé

Returns

le message

4.6.3.5 char* findRoomFromSocket (int sock)

Parcourt chaque room à la recherche du socket, et renvoie la room qui correspond.

findRoomFromSocket

Parameters

<i>sock</i>	le socket a chercher
-------------	----------------------

Returns

char* room, la room où se trouve le socket

4.6.3.6 char* getServerResponse (char * commandLine)

analyse la commande du client pour agir en fonction

getServerResponse

Parameters

<i>commandLine</i>	la commande utilisateur entrée par le client
--------------------	--

Returns

la reponse du serveur qui correspond

4.6.3.7 int incrementInsult (sockaddr_in adresse)

incremente le nombre d'insultes d'un client, et le kick si supérieur à 15 a chaque message

incrementInsult

Parameters

<i>adresse</i>	l'ip du client
----------------	----------------

Returns

1 si l'incrementation est effectuée, -1 sinon

4.6.3.8 main (int argc, char ** argv)

main, ouvre le serveur et se met en attente de connexions

Parameters

<i>argc</i>	
<i>argv</i>	

Returns

1 si echec, 0 sinon

main, ouvre le serveur et se met en attente de connexions

main

Parameters

<i>argc</i>	
<i>argv</i>	

Returns

1 en cas d'echec, 0 sinon

4.6.3.9 void readWords (dictionary * d)

Ouvre le fichier de mots et lit chaque ligne pour les stocker dans le dictionnaire.

readWords

Parameters

<i>d</i>	le dictionnaire a remplir
----------	---------------------------

Returns

void

4.6.3.10 int removeRoom (char * roomName)

Retire la room désirée, et décale toutes les autres dans la structure.

removeRoom

Parameters

<i>roomName</i>	le nom de la room a supprimer
-----------------	-------------------------------

Returns

1 si la suppression est reussie, 0 sinon

4.6.3.11 int removeSocketFromRoom (int sock, char * roomName)

Supprime le socket d'une room et decalle tous les autres.

removeSocketFromRoom

Parameters

<i>sock</i>	le socket a supprimer
<i>roomName</i>	la room ou se trouve le socket

Returns

1 si suppression reussie, 0 sinon.

4.6.3.12 void* renvoi_message (void * *arg*)

Boucle de reception et envoi du message a la room.

renvoi_message

Parameters

<i>arg</i>	le message envoyé par le client
------------	---------------------------------

Returns

void

4.6.3.13 void* sendMessageToRoom (void * *rMsg*)

Fonction threadee d'envoi de message a tous les sockets d'une room.

sendMessageToRoom

Parameters

<i>rMsg</i>	le message a envoyer
-------------	----------------------

Returns

void

4.6.3.14 void stop ()

Ferme le server après avoir fermé toutes les room.

stop

Returns

void

Ferme le server après avoir fermé toutes les room.

[stop\(\)](#)

Returns

void

4.6.3.15 int updateUserFile (sockaddr_in *adresse*)

vérifie la présence d'un utilisateur dans le fichier client

updateUserFile

Parameters

<i>adresse</i>	l'ip du client
----------------	----------------

Returns

retourne 1 si l'utilisateur est nouveau, 0 si déjà connu, -1 si erreur

Index

- dictionary, [5](#)
 - sz, [5](#)
 - words, [5](#)
- name
 - users, [6](#)
- room
 - rooms, [6](#)
- rooms, [6](#)
 - room, [6](#)
 - sz, [6](#)
- socks
 - users, [6](#)
- sz
 - dictionary, [5](#)
 - rooms, [6](#)
 - users, [6](#)
- users, [6](#)
 - name, [6](#)
 - socks, [6](#)
 - sz, [6](#)
- words
 - dictionary, [5](#)