Protocole

Generated by Doxygen 1.8.10

Thu Oct 27 2016 21:42:26

Contents

Chapter 1

projet creation de protocole

Vous trouverez ici toute la documentation relative au projet de création d'un protocole client serveur.

Author

Gregory GUEUX gregory.gueux@etu.univ-lyon1.fr

1.1 le projet

1.1.1 En résumé

le but du projet est de créer un protocole de communication entre deux processus qui n'ont pas forcément de lien entre eux (via un fork par exemple). Cela, grâce à l'utilisation des sockets.

1.1.2 Client

Le client est le programme qui doit se trouver sur la machine de l'utilsateur. Il permet d'envoyer un rapport au Serveur ou de demander que ce dernier fasse un rapport completer sur les derniers rapport reçu.

1.1.3 Serveur

Le serveur doit être présent soit sur une machine distante soit sur la machine en local. Il s'occupe de récupèrer les rapports des clients et de faire un rapport sur ces derniers sur demande du client.

Warning

il doit être lancé AVANT le Client.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Module Documentation

5.1 Client

Les fonctions principaux du client.

Data Structures

- struct sbufferToSend
- struct sBufferToSend

  Représente un rapport complet avec le header à envoyer.

Typedefs

- typedef struct sbufferToSend **bufferToSend**

Functions

- void menu ()

  affichie le menu des choix à l'écran.
- void receiveRapport ()

  permet de recevoir le rapport mensuelle.
- void newReport ()

  Permet d'écrire ou d'envoyer un nouveau rapport.
- void sendReport (int socketServeur, bufferToSend ∗report)

  envoie le rapport au seveur.
- void kingdomOfFreedom (void ∗∗newFreeGuys)

  parce que tout le monde aspire à la liberté, même des octets veulent pouvoir bénéficier de cela.
- void pseudoUser (char pseudo[25])

  Permet de récupérer le pseudo de l'utilisateur sur l'entrée standard.
- void set_buffer_struct (bufferToSend ∗report)

  Complète la structure du rapport. C'est-à-dire la taille du pseudo et la taile du message.
- void sendReportToServeur (const int sServeur, bufferToSend ∗report)

  Envoie l'ordre au client.
- void stopServeur ()

  Arrête le serveur distant.
- void sendToServeur (int sServeur, char ∗msg, int sizeMsg)

  envoie quelque chose sur la socket donnée

Variables

- char order [4]

  l'ordre qui est envoyé.
- char sizePseudo [3]

  La taille du pseudo.
- char pseudo [25]

  Le pseudo.
- char sizeMsg [256]

  La taille du message. Qui est limité à 10^ 256 caractère maxi.
- char * msg

  le message.
- void(*)() menuChosen ()

  La fonction permet de selection une fonction selon le choix du client.

5.1.1 Detailed Description

Les fonctions principaux du client.

5.1.2 Function Documentation

5.1.2.1 void kingdomOfFreedom (void ** *newFreeGuys*)

parce que tout le monde aspire à la liberté, même des octets veulent pouvoir bénéficier de cela.

Parameters

newFreeGuys	Un candidiat pour rejoindre le royaume.

5.1.2.2 void pseudoUser (char *pseudo[25]*)

Permet de récupérer le pseudo de l'utilisateur sur l'entrée standard.

Parameters

pseudo	

5.1.2.3 void sendReport (int *socketServeur,* **bufferToSend** * *report*)

envoie le rapport au seveur.

Parameters

socketServeur	La socket du serveur.
report	La structure

5.1.2.4 void sendReportToServeur (const int *sServeur,* **bufferToSend** * *report*)

Envoie l'ordre au client.

Parameters

sServeur	La socket qui est connecté au client.
report	La structure du rapport.

5.1.2.5 void sendToServeur (int *sServeur,* char * *msg,* int *sizeMsg*)

envoie quelque chose sur la socket donnée

Parameters

sServeur	La socket du serveur
msg	Le message à envoyer
sizeMsg	La taille du message à envoyer.

5.1.2.6 void set_buffer_struct (**bufferToSend** * *report*)

Complète la structure du rapport. C'est-à-dire la taille du pseudo et la taile du message.

Parameters

report	Le rapport complet.

5.1.3 Variable Documentation

5.1.3.1 void(*)() menuChosen()

La fonction permet de selection une fonction selon le choix du client.

Returns

Le pointeur de la fonction choisie par le client.

5.2 LibSock

Les fonctions utiles pour la communications entre le client et le serveurs.

Functions

- int CreeSocketServeur (const char ∗port)

  Cree une socket d'attente pour le serveur sur le port port.
- int **CreeSocketClient** (const char ∗serveur, const char ∗port)
- int AcceptConnexion (int s)

  retourne la socket accepte
- int RecoieEtSauveDonnees (int fd, int sock)
- char ∗ RecoieLigne (int sock)
- int EnvoieMessage (int s, char ∗format,...)
- int TestLecture (int s)

5.2.1 Detailed Description

Les fonctions utiles pour la communications entre le client et le serveurs.

5.2.2 Function Documentation

5.2.2.1 int CreeSocketServeur (const char ∗ *port*)

Cree une socket d'attente pour le serveur sur le port port.

Parameters

port	: le port utilisé

Returns

la socket d'attente ou -1 en cas d'erreur

en cas d'erreur, un message explicatif est affich sur la sortie d'erreur standart

5.2.2.2 int EnvoieMessage (int *s*, char ∗ *format*, ...)

envoie le message format sur la socket s (comme un printf)

Parameters

s	: la socket sur laquel ecrire le message
format	: le format du message (comme pour printf)
...	

Returns

le nombre d'octet crit ou -1 s'il y a eu un problme

5.2.2.3 int RecoieEtSauveDonnees (int *fd*, int *sock*)

Lire les donnes sur une socket et les crire automatiquement dans un descripteur de fichier Cette fonction stoppe la lecture lorsque la soket est ferme.

Parameters

sock	: la socket d'o proviennent les donnes, cela fonctionne aussi si sock est un file descriptor quelconque
fd	: le descripteur de fichier sur lesquel enregistrer les donnes

Returns

: si cela a fonctionn le nombre d'octets lus, sinon -1

5.2.2.4 char* RecoieLigne (int *sock*)

Lire les données sur une socket (uniquement j'utilise recv) jusqu'à arriver à un retour chariot '
' la donnée est stokée dans un tableau dont la taille est adapté pour cela).

Parameters

sock	: la socket de lecture

Returns

la chaine lue à libérer par free

5.2.2.5 int TestLecture (int *s*)

Regarde s'il y a qqchose lire sur une socket

Parameters

s	: la socket

Returns

* -1 s'il y a une erreur
  - -2 s'il n'y a rien lire
  - 0 si la socket est ferme
  - 1 s'il y a qqchose lire

Regarde s'il y a qqchose à lire sur une socket

Parameters

s	: la socket

Returns

* -1 s'il y a une erreur
  - -2 s'il n'y a rien à lire
  - 0 si la socket est fermée
  - 1 s'il y a qqchose à lire

5.3 UsualFonction

Les fonctions utile pour le client et le serveur.

Functions

- void fctErrorPersonal (int value)

  evalue la fonction si elle ne retourne pas une erreur.
- int readLine (char ∗str, const int size)

  lit une ligne sur l'entrée standard et vérifier la fonction fgets.
- int readChar (char ∗c)

  lit un caractère sur l'entrée standard et vérifier la fonction fgetc.
- void viderBuffer ()

  vide le buffer de la sortie standard.
- void convertIntToChar (char ∗tab, const size_t intergerToConvert)

  Convertie un entier int en char ∗.

5.3.1 Detailed Description

Les fonctions utile pour le client et le serveur.

5.3.2 Function Documentation

5.3.2.1 void convertIntToChar (char ∗ *tab,* const size_t *intergerToConvert*)

Convertie un entier int en char ∗.

Parameters

tab	le tableau ou sera stocker le char∗
intergerTo↩ Convert	L'entier à convertir.

5.3.2.2 void fctErrorPersonal (int *value*)

evalue la fonction si elle ne retourne pas une erreur.

Warning

si une erreur est détecté, le programme est quittée sans fermer le serveur.

Parameters

value	la valeur du param retourné.

5.3.2.3 int readChar (char ∗ *c*)

lit un caractère sur l'entrée standard et vérifier la fonction fgetc.

Parameters

c	Un pointeur sur un char

Returns

0 si fgetc a rencontré une erreur.
1 si tout s'est bien passé.

5.3.2.4 int readLine (char * *str*, const int *size*)

lit une ligne sur l'entrée standard et vérifier la fonction fgets.

Parameters

str	le buffer où sera écrit la chaine de caractère.
size	la taille à lire.

Returns

0 si fgets n'a pas fonctionné.
1 si tout s'est bien passsé.

5.4 Serveur

Les fonctions principals du serveur.

Data Structures

- struct sbufferToReceive
- struct sBufferToSend

  Représente un rapport complet avec le header à envoyer.

Typedefs

- typedef struct sbufferToReceive **bufferToReceive**

Functions

- void recupOrder (const int sClient, char ∗buff)

  récupère l'order du client.
- int newCommunication (int sClient)

  Initialise une nouvelle communication entre le client et le serveur.
- void waitConnexion (int s)

  Permet de mettre le processus sur écoute et d'attendre une connexion entrante.
- void creationSocket ()

  Crée un socket pour le serveur.
- void newReport (int sClient, bufferToReceive ∗report)

  récupère et enregistre le rapport qu'a envoyé le client.
- void newSendReport (int sClient, bufferToReceive ∗report)

  envoie tout les rapports du clients dans un seul fichier.
- void recupTime (char ∗newDateAndTime, time_t time)

  Renvoie la date Local + l'heure local.
- void addLog (bufferToReceive ∗report, const char ∗msg, const char ∗prefixe)

  Permet d'ajouter un entrée dans le fichier de log de la sessions.
- void recupInformation (int sClient, bufferToReceive ∗report)

  récupère les informations qui sont forcément envoyer.

Variables

- int order

  l'ordre qui est envoyé.
- int sizePseudo

  La taille du pseudo.
- char pseudo [25]

  Le pseudo.
- int sizeMsg

  La taille du message.
- char ∗ msg

  le message.
- char time [256]

  Le temps pour les log.
- FILE ∗ log

  le fichier de log.

5.4.1 Detailed Description

Les fonctions principals du serveur.

5.4.2 Function Documentation

5.4.2.1 void addLog (bufferToReceive * *report*, const char * *msg*, const char * *prefixe*)

Permet d'ajouter un entrée dans le fichier de log de la sessions.

Parameters

report	La structure qui contient le fd du fichier de log.
msg	le message à mettre dans le fichier de log.
prefixe	Le préfixe à mettre avant le message.

5.4.2.2 int newCommunication (int *sClient*)

Initialise une nouvelle communication entre le client et le serveur.

Note

On procéde a un fork pour permettre au serveur de continuer à attendre des communications.

Parameters

sClient	La sockets du client.

Returns

5.4.2.3 void newReport (int *sClient*, bufferToReceive * *report*)

récupère et enregistre le rapport qu'a envoyé le client.

Parameters

sClient	La socket du client.
report	Le rapport du client.

5.4.2.4 void newSendReport (int *sClient*, bufferToReceive * *report*)

envoie tout les rapports du clients dans un seul fichier.

Parameters

sClient	La socket du client.
report	Le rapport du client.

5.4.2.5 void recupInformation (int *sClient*, bufferToReceive * *report*)

récupère les informations qui sont forcément envoyer.

Parameters

sClient	La socket du client.
report	La structure à remplir.

5.4.2.6 void recupOrder (const int *sClient*, char * *buff*)

récupère l'order du client.

Parameters

sClient	La socket du client.
buff	Le buffer qui contiendra l'order.

5.4.2.7 void recupTime (char * *newDateAndTime*, time_t *time*)

Renvoie la date Local + l'heure local.

Parameters

newDateAndTime	La chaine de caractère où stocker la date.
time	La date réinitialiser.

5.4.2.8 void waitConnexion (int *s*)

Permet de mettre le processus sur écoute et d'attendre une connexion entrante.

Parameters

s	La socket qui est disponible sur un port. donnée.

Chapter 6

Data Structure Documentation

6.1 sbufferToReceive Struct Reference

Data Fields

- int order

  l'ordre qui est envoyé.
- int sizePseudo

  La taille du pseudo.
- char pseudo [25]

  Le pseudo.
- int sizeMsg

  La taille du message.
- char * msg

  le message.
- char time [256]

  Le temps pour les log.
- FILE * log

  le fichier de log.

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

- /Users/gregorygueux/ClionProjects/ASR5_MiniProjet/src/serveur/headerServeur.h

6.2 sBufferToSend Struct Reference

Représente un rapport complet avec le header à envoyer.

`#include <headerClient.h>`

6.2.1 Detailed Description

Représente un rapport complet avec le header à envoyer.

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

- /Users/gregorygueux/ClionProjects/ASR5_MiniProjet/src/client/headerClient.h

6.3 sbufferToSend Struct Reference

Data Fields

- char order [4]

  l'ordre qui est envoyé.
- char sizePseudo [3]

  La taille du pseudo.
- char pseudo [25]

  Le pseudo.
- char sizeMsg [256]

  La taille du message. Qui est limité à 10^ 256 caractère maxi.
- char * msg

  le message.

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

- /Users/gregorygueux/ClionProjects/ASR5_MiniProjet/src/client/headerClient.h

Chapter 7

File Documentation

7.1 /Users/gregorygueux/ClionProjects/ASR5_MiniProjet/src/client/headerClient.h File Reference

Contient les prototypes des fonctions de main.c.

```
#include <stdlib.h>
#include <stdio.h>
#include "../commun/socklib.h"
#include "../commun/usualFct.h"
```

Data Structures

- struct sbufferToSend

Macros

- #define **SEND_RAPPORT** "001"
- #define **MAKE_MRAPPORT** "002"
- #define **STOP_SERVEUR** "003"
- #define **HOST** "localhost"
- #define **PORT** "8080"
- #define **TAILLE_MSG** 32768

Typedefs

- typedef struct sbufferToSend **bufferToSend**

Functions

- void menu ()

  affichie le menu des choix à l'écran.
- void receiveRapport ()

  permet de recevoir le rapport mensuelle.
- void newReport ()

  Permet d'écrire ou d'envoyer un nouveau rapport.
- void sendReport (int socketServeur, bufferToSend *report)

envoie le rapport au seveur.

- void kingdomOfFreedom (void **newFreeGuys)

  parce que tout le monde aspire à la liberté, même des octets veulent pouvoir bénéficier de cela.
- void pseudoUser (char pseudo[25])

  Permet de récupérer le pseudo de l'utilisateur sur l'entrée standard.
- void set_buffer_struct (bufferToSend *report)

  Complète la structure du rapport. C'est-à-dire la taille du pseudo et la taile du message.
- void sendReportToServeur (const int sServeur, bufferToSend *report)

  Envoie l'ordre au client.
- void stopServeur ()

  Arrête le serveur distant.
- void sendToServeur (int sServeur, char *msg, int sizeMsg)

  envoie quelque chose sur la socket donnée

Variables

- void(*)() menuChosen ()

  La fonction permet de selection une fonction selon le choix du client.

7.1.1 Detailed Description

Contient les prototypes des fonctions de main.c.

Author

Grégory Gueux gregory.gueux@etu.univ-lyon1.fr

7.2 /Users/gregorygueux/ClionProjects/ASR5_MiniProjet/src/commun/socklib.h File Reference

Contient les prototypes des fonctioncs de socklib.c.

```
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <stdio.h>
```

Macros

- #define **INVALID_SOCKET** -1
- #define **SOCKET_ERROR** -1

Functions

- int CreeSocketServeur (const char *port)

  Cree une socket d'attente pour le serveur sur le port port.
- int **CreeSocketClient** (const char *serveur, const char *port)
- int AcceptConnexion (int s)

  retourne la socket accepte

- int RecoieEtSauveDonnees (int fd, int sock)
- char * RecoieLigne (int sock)
- int EnvoieMessage (int s, char *format,...)
- int TestLecture (int s)

7.2.1 Detailed Description

Contient les prototypes des fonctioncs de socklib.c.

Author

Rico Fabien fabien.rico@univ-lyon1.fr

7.3 /Users/gregorygueux/ClionProjects/ASR5_MiniProjet/src/commun/usualFct.h File Reference

Contient les prototypes des fonctions de usualFct.c.

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

Functions

- void fctErrorPersonal (int value)

  evalue la fonction si elle ne retourne pas une erreur.
- int readLine (char *str, const int size)

  lit une ligne sur l'entrée standard et vérifier la fonction fgets.
- int readChar (char *c)

  lit un caractère sur l'entrée standard et vérifier la fonction fgetc.
- void viderBuffer ()

  vide le buffer de la sortie standard.
- void convertIntToChar (char *tab, const size_t intergerToConvert)

  *Convertie un entier int en char *.*

7.3.1 Detailed Description

Contient les prototypes des fonctions de usualFct.c.

Author

Grégory GUEUX gregory.gueux@etu.univ-lyon1.fr