

PROJET LO41

FORUM DE DISCUSSION

Mickaël Barroux

Philippe Chen

SOMMAIRE

1	Cahier des charges.....	3
1.1	Présentation générale.....	3
1.2	Description de la structure de la base de données.....	4
1.3	Description de la communication entre le client et le serveur.....	5
1.4	Description du serveur.....	7
1.5	Description du client.....	7
2	Dossier de réalisation.....	8
3	Documentation.....	10
3.1	Référence du fichier fonctions_client.c.....	10
3.1.1	Fonctions.....	10
3.1.2	Description détaillée.....	11
3.1.3	Documentation des fonctions.....	11
3.2	Référence du fichier fonctions_serveur.c.....	17
3.2.1	Fonctions.....	17
3.2.2	Description détaillée.....	18
3.2.3	Documentation des fonctions.....	19

1 CAHIER DES CHARGES

1.1 Présentation générale

Un forum de discussion est un lieu d'échange et de partage d'informations.

Le forum est principalement basé sur un serveur et les utilisateurs y accèdent à partir d'un client, par exemple, pour le cas des forums Internet, le serveur est souvent un serveur HTTP et le client est un simple navigateur web.

Dans notre cas, nous allons réutiliser ce concept avec un client / serveur codé en C.

Le serveur possèdera la base de données du forum, les clients se connecteront au serveur et lui enverront des requêtes auquel le serveur va répondre. Le client traitera ensuite la réponse et l'affichera à l'utilisateur.

Le forum permettra les actions suivantes :

- Gestion des utilisateurs
 - Création d'un nouveau compte
 - Connexion d'un utilisateur
 - Récupération d'un mot de passe perdu
- Sujets
 - Lister les sujets
 - Créer un nouveau sujet
- Messages
 - Lire les messages d'un sujet

- Ecrire un nouveau message dans un sujet
- Modifier un message posté précédemment
- Le modérateur du forum possède des droits supplémentaires
 - Modifier n'importe quel sujet ou message
 - Supprimer n'importe quel sujet ou message
 - Verrouiller un sujet (le rendre disponible en lecture seule uniquement)
- Abonnements à un sujet
 - S'abonner à un sujet pour recevoir une notification de l'arrivée de nouveaux messages sur le sujet en question
 - Se désabonner d'un sujet
 - Liste des abonnements d'un utilisateur
- Log : reporte dans un fichier de log les requêtes reçues :
 - Connexion d'un utilisateur
 - Création de nouveau sujet/message
 - Modérations sur un sujet/message

1.2 Description de la structure de la base de données

Nous avons décidé, après réflexion, d'utiliser l'architecture du système de gestion de fichiers d'UNIX pour représenter la base de données du forum.

La base de données se situera alors dans un répertoire du serveur qui devra donc avoir accès à des droits de création, d'écriture et de lecture sur ce dernier. Il y aura donc un répertoire « forum » avec les différents sujets dedans.

Chaque sujet sera représenté par un fichier de type texte, lequel contiendra:

- Le statut du sujet (verrouillé ou non)
- La liste des utilisateurs abonnés au sujet
- La liste des messages appartenant au sujet avec l'identifiant du créateur, la date de création et l'id du message

Les utilisateurs seront quant à eux représentés dans un fichier texte « user » (à la façon du fichier /etc/passwd sous UNIX) qui définira :

- L'identifiant de l'utilisateur
- Son mot de passe en clair (il est envisagé de crypter son mot de passe dans une version ultérieure)
- Son adresse e-mail utilisée pour l'avertir lors de notifications sur des sujets auxquels il est abonné ou en cas d'oubli de mot de passe
- La liste des sujets auxquels il est abonné
- Le type de compte (Modérateur ou simple utilisateur)

Chaque action sera donc reportée dans un fichier « log » qui enregistrera toutes les requêtes reçues par le serveur.

Dans une amélioration ultérieure du système, nous envisagerons de gérer des catégories ou des sous-forums en les représentant par des sous-répertoires du répertoire « forum ».

1.3 Description de la communication entre le client et le serveur

Nous avons opté pour une communication par socket entre le client et le serveur en mode connecté (protocole TCP) pour ainsi permettre une connexion à distance au forum et non uniquement sur la machine locale.

La communication se fait donc en mode connecté : une fois un client connecté, le serveur se met alors en attente de requêtes sous forme de messages. Le serveur traite alors ces requêtes et renvoi une réponse au client qu'il devra interpréter en fonction d'un code de retour.

Voici dans un premier temps les requêtes possibles du client :

LOGIN identifiant:mot_de_passe	DELMESSAGE nom_du_sujet:id_message
ADDUSER identifiant:mot_de_passe:email	DELTOPIC nom_du_sujet
OUBLI identifiant:email	ABOTOPIC nom_du_sujet
READ nom_du_sujet	LIST
NEWTOPIC nom_du_sujet	EXIT
NEWMESSAGE nom_du_sujet	LISTERABO
NEWMESSAGE nom_du_sujet:message	DESABO
EDITMESSAGE nom_du_sujet:id_message:message	LOCK nom_du_topic
TOPICNAME id_topic	UNLOCK nom_du_topic

La liste est soumise à modifications ultérieures selon les versions du projet

Afin d'apporter plus de souplesse au projet, il est possible de spécifier l'adresse et le port du serveur auquel le client doit se connecter par des options sur la ligne de commande, pour ainsi avoir accès à plusieurs forums différents avec le même client. Il est également possible de changer le port d'écoute du serveur par une option de la ligne de commande.

- Client :
 - serveur par défaut : localhost (serveur sur machine locale)
 - port par défaut : 65000
 - spécifier une nouvelle adresse de serveur : -h adresse_serveur
 - spécifier un nouveau port de serveur : -p port_serveur
- Serveur :
 - port par défaut 65000

- spécifier un autre port d'écoute : -p port_ecoute

1.4 Description du serveur

Le serveur est là pour interagir avec la base de données et les différents clients qui se connectent à lui. Pour pouvoir gérer plusieurs clients, nous avons choisi de mettre en place des « threads », processus légers, qui seront créés lors de la connexion d'un client et détruit lors de sa déconnexion.

Lors de chaque connexion d'un client sur le serveur, ce dernier lance un thread qui s'occupera de gérer toutes les requêtes du client qui lui a été attribué.

Chaque processus doit alors pouvoir accéder à la base de données, lire et écrire dessus sans entrer en conflit, d'où l'implémentation de mutex pour gérer l'accès aux zones critiques.

1.5 Description du client

Le client doit principalement gérer l'aspect affichage pour l'utilisateur. Ce dernier doit pouvoir jouir d'une utilisation aisée avec un affichage clair et intuitif (du moins autant que possible en mode console :p)

Nous avons donc choisi une navigation simple en associant toutes les commandes à des numéros que l'utilisateur devra entrer selon le point de navigation où il est situé (menus).

Lors de l'exécution du client, l'utilisateur dispose alors d'un menu d'accueil lui permettant diverses actions comme : l'inscription au forum, l'identification s'il est déjà inscrit ou la possibilité de retrouver son mot de passe si celui-ci a été oublié.

Après cette étape, le client doit alors être connecté pour pouvoir accéder au menu principal, ce menu permet entre autre de : lister les sujets existants, de consulter un sujet précis, ou d'en démarrer un nouveau. Dans le cas où l'utilisateur est un modérateur, il doit aussi pouvoir supprimer un sujet. L'option de gestion des abonnements sera aussi disponible sur ce menu.

Enfin, le dernier menu disponible est celui de la consultation d'un sujet : tous les messages sont alors automatiquement listés pour la lecture. Entre autre, le menu offre la possibilité de : poster un nouveau message, d'éditer un de ses propres message existant ou de s'abonner au sujet, une option pour effacer un message sera disponible pour les modérateurs.

2 DOSSIER DE RÉALISATION

Nous nous sommes rendus compte, au cours du temps, que le cahier des charges était un peu exhaustif et que le projet un peu ambitieux. De plus, le projet reposait sur de la gestion de fichiers en C ce qui n'était pas véritablement le but du projet de l'UV de LO41. Nous avons donc été contraint, devant un manque cruel de temps, de réduire les fonctionnalités du projet tout en assurant son fonctionnement de base et en se concentrant notamment sur les notions vues pendant l'UV de LO41, c'est à dire pour ce projet : les sockets pour connecter un client à un serveur, le serveur multi-threads qui permet de générer un processus serveur (thread) pour chaque client qui se connecte à lui et également les mutex qui permettent d'assurer l'accès protégé aux ressources critiques du projet (sujets, fichier de log, fichier d'utilisateurs) par les différents clients.

En ce qui concerne le développement, les fichiers sources ont été décomposés afin de pouvoir se partager les différentes fonctions du projet entre les deux membres du groupe et permettre une gestion simplifiée du code source.

En effet, nous avons utilisé deux fichiers d'en-têtes (.h) : un pour le client et un autre pour le serveur, dans lesquels nous avons mis les librairies nécessaires au projet (*#include*), les macros permettant une modification aisée du code en ce qui concerne les valeurs fixes (*#define*) et les squelettes des fonctions développées dans le source du client et du serveur.

Deux fichiers (.c) supplémentaires ont été utilisés afin de recenser les fonctions utilisées dans le serveur et dans le client. Ces fichiers sont nommés *fonctions_serveur.c* et *fonctions_client.c*

Enfin, les fichiers de base du client et du serveur sont nommés *client.c* et *serveur.c* dans lesquels on lance les fonctions de bases définies dans les fichiers relatifs aux fonctions du client et du serveur.

Le code source a été commenté avec soin tout au long du projet, à l'aide de balises spécifiques utilisées afin de pouvoir générer par la suite une documentation complète du projet à l'aide de Doxygen. Cette documentation est disponible ci-après, ou bien en HTML avec le code source du projet qui vous a été transmis par mail.

Exemple :

```
/**
 *  \fn int Connecter_Client(char *Nom_Machine, int port, struct
 sockaddr_in *Nom_Socket)
 *      Cette fonction permet l'ouverture au niveau TCP d'une
 connexion entre un client et un serveur via un socket
 *  \param Nom_Machine : Nom du serveur distant (IP sur serveur)
 *  \param port : Port du serveur distant
 *  \retval Nom_Socket est une structure de type sockaddr_in
 contenant les informations sur le socket sur lequel le client s'est
 connecté
 *  \return Le descripteur du socket auquel le client s'est
 connecté (-1 si erreur)
 */
```

Un script de compilation en Bash a été utilisé afin de compiler dans le bon ordre et avec les bonnes options tous ces fichiers, étant donné que les sources du projet étaient décomposées. Ce script a été légèrement modifié lors du passage du développement sous Ubuntu au développement sous Solaris (notamment les bibliothèques à inclure lors de la compilation et l'édition de liens). La création d'un makefile avait été envisagé mais abandonné faute de temps...

Script compil.sh :

```
#!/bin/sh
gcc -c fonctions_client.c
gcc -c fonctions_serveur.c
gcc -o client client.c fonctions_client.o -lnsl -lsocket
gcc -D_REENTRANT -o serveur serveur.c fonctions_serveur.o -lpthread
-lnsl -lsocket
cp serveur ../forum/
```

3 DOCUMENTATION

3.1 Référence du fichier fonctions_client.c

Fonctions de base d'utilisation des sockets. [Plus de détails...](#)

```
#include "fonctions_client.h"
```

3.1.1 Fonctions

int [Connecter_Client](#) (char *Nom_Machine, int port, struct sockaddr_in *Nom_Socket)

void [Fermer](#) (int desc)

int [Emettre](#) (int desc, char *message)

int [Recevoir](#) (int desc, char *tampon, int longueur_tampon)

void **purger** (void)

void **clean** (char *chaine)

void **clrscr** ()

int [car_autorise](#) (char *chaine)

void [accueil](#) (int desc)

void [inscription](#) (int desc)

void [connexion](#) (int desc)

void [oubli](#) (int desc)

void [quitter_client](#) (int desc)

void [menu](#) (int desc)

void [getTopicName](#) (int desc, int id, char *name)

void [liste](#) (int desc)

void [consulter](#) (int desc)

void [new_topic](#) (int tmp)

```
void del\_topic (int tmp)
void new\_message (int tmp, char *topic_name)
void menu\_consultation (int tmp, char *topic_name)
void lire\_message (int tmp, char *topic_name)
void abo\_topic (int tmp, char *topic_name)
```

3.1.2 Description détaillée

Fonctions de base d'utilisation des sockets.

Auteur:

Mickaël Barroux & Philippe Chen

Version:

1.0

Date:

28 novembre 2008

3.1.3 Documentation des fonctions

```
void abo_topic ( int    tmp,
                char *  topic_name
                )
```

Cette fonction permet à l'utilisateur de s'abonner au sujet

Paramètres:

tmp : descripteur de socket pour la connexion

topic_name : nom du sujet concerne

```
void accueil ( int desc )
```

Cette fonction affiche a l'écran la page d'accueil du forum L'utilisateur peut acceder de cette page aux pages suivantes: 1. Inscription 2. Connexion 3. Oubli de mot de passe

Paramètres:

desc : descripteur de socket pour la connexion

```
int car_autorise ( char * chaine )
```

Cette fonction permet de tester si la chaîne contient des caractères spéciaux

Paramètres:

chaine : Chaîne de caractères à tester

Renvoie:

0 si la chaîne contient un caractère non autorisé ou 1 sinon

```
int Connecter_Client ( char * Nom_Machine,  
int port,  
struct sockaddr_in * Nom_Socket  
)
```

Cette fonction permet l'ouverture au niveau TCP d'une connexion entre un client et un serveur via un socket

Paramètres:

Nom_Machine : Nom du serveur distant (IP sur serveur)

port : Port du serveur distant

Valeurs retournées:

Nom_Sock est une structure de type `sockaddr_in` contenant les informations sur le
et socket sur lequel le client s'est connecté

Renvoi:

Le descripteur du socket auquel le client s'est connecté (-1 si erreur)

```
void connexion ( int desc )
```

Cette fonction affiche a l'écran la page de connexion au forum L'utilisateur peut tenter de s'identifier au forum en fournissant les informations suivantes: Identifiant, mot de passe En cas d'oubli du mot de passe, il peut acceder a la page: Oubli du mot de passe après chaque tentative erronée.

Paramètres:

desc : descripteur de socket pour la connexion

```
void consulter ( int desc )
```

Cette fonction affiche la liste des messages d'un sujet

Paramètres:

desc : descripteur de socket pour la connexion

```
void del_topic ( int tmp )
```

Cette fonction demande la suppression d'un sujet

Paramètres:

tmp : descripteur de socket pour la connexion

```
int Emettre ( int desc,  
             char * message  
            )
```

Cette fonction permet d'émettre des caractères depuis une connexion identifiée par le descripteur de socket *desc*

Paramètres:

desc : descripteur de socket identifiant la connexion

message : tableau de caractères à émettre

Renvoie:

-1 si erreur de transmission

```
void Fermer ( int desc )
```

Cette fonction permet la fermeture de la connexion désignée par le descripteur de socket *desc*

Paramètres:

desc : descripteur de socket identifiant la connexion

```
void getTopicName ( int desc,  
                  int id,  
                  char * name  
                  )
```

Cette fonction demande le nom d'un sujet en fonction d'un id qui correspond a la position du sujet dans la liste des sujets

Paramètres:

desc : descripteur de socket pour la connexion

id : id du topic

name : nom du topic

```
void inscription ( int desc )
```

Cette fonction affiche a l'écran la page d'inscription du forum L'utilisateur peut ici s'inscrire sur le forum en fournissant les informations suivantes: Identifiant, mot de passe, e-mail.

Paramètres:

desc : descripteur de socket pour la connexion

```
void lire_message ( int    tmp,  
                  char *  topic_name  
                  )
```

Cette fonction affiche a l'écran les messages d'un sujet

Paramètres:

tmp : descripteur de socket pour la connexion

topic_name : nom du sujet concerne

```
void liste ( int  desc )
```

Cette fonction affiche la liste des sujets

Paramètres:

desc : descripteur de socket pour la connexion

```
void menu ( int  desc )
```

Cette fonction affiche a l'écran la page du menu principal du forum L'utilisateur peut acceder de cette page aux pages suivantes: 1. Liste des sujets 2. Consultation de sujet 3. Creation de sujet Si l'utilisateur possede les droits de modérateur du forum, il pourra aussi: 4. Supprimer un sujet

Paramètres:

desc : descripteur de socket pour la connexion

```
void menu_consultation ( int    tmp,  
                        char *  topic_name  
                        )
```

Cette fonction affiche a l'écran la page de consultation d'un sujet

Paramètres:

tmp : descripteur de socket pour la connexion

topic_name : nom du sujet concerne

```
void new_message ( int    tmp,  
                 char *  topic_name  
                 )
```

Cette fonction demande la création un nouveau message sur le sujet associe

Paramètres:

tmp : descripteur de socket pour la connexion

topic_name : nom du sujet ou sera poste le message

```
void new_topic ( int  tmp )
```

Cette fonction demande la création d'un nouveau sujet

Paramètres:

tmp : descripteur de socket pour la connexion

```
void oubli ( int  desc )
```

Cette fonction affiche a l'écran la page d'oubli du mot de passe L'utilisateur peut ainsi demander la reemission de son mot de passe en fournissant les informations suivantes:

Identifiant, e-mail

Paramètres:

desc : descripteur de socket pour la connexion

```
void quitter_client ( int desc )
```

Cette fonction permet à un client de se déconnecter du serveur de forum

Paramètres:

desc descripteur du socket

```
int Recevoir ( int desc,  
              char * tampon,  
              int longueur_tampon  
              )
```

Cette fonction permet de recevoir des caractères depuis une connexion identifiée par le descripteur de socket *desc*

Paramètres:

desc : descripteur de socket identifiant la connexion

longueur_tampon : donne la longueur maximum du tampon de réception

Valeurs retournées:

tampon est le tableau de caractères contenant les caractères reçus

Renvoie:

Le nombre de caractères effectivement reçus (-1 si erreur)

3.2 Référence du fichier fonctions_serveur.c

Fonctions de base d'utilisation des sockets. [Plus de détails...](#)

```
#include "fonctions_serveur.h"
```

3.2.1 Fonctions

int [str_istr](#) (const char *cs, const char *ct)

void [neutraliserEspaces](#) (char *chaine)

int [Creer_Serveur](#) (int port, struct sockaddr_in *Nom_Socket)

int [Serveur_Attendre_Client](#) (int descripteur_socket)

void * [thread_client](#) (void *p_data)

void [Fermer](#) (int desc)

int [Emettre](#) (int desc, char *message)

int [Recevoir](#) (int desc, char *tampon, int longueur_tampon)

void [VerifConnexion](#) (int desc, char *question)

void [adduser](#) (int desc, char *question)

void [RandomString](#) (char *str, size_t n)

void [oubli_pass](#) (int desc, char *question)

void [recup_id_topic](#) (int desc, int id)

void [lister_messages_sujet](#) (int desc, char *nom_sujet)

int [recup_id_message_max](#) (char *nom_sujet)

void [ListeTopics](#) (int desc)

void [NouveauSujet](#) (int desc, char *nom_sujet)

void [SupprimerSujet](#) (int desc, char *nom_sujet)

void [NouveauMessage](#) (int desc, char *user, char *topic_name, char *msg)

void [EcritureLog](#) (char *operation)

void [Abonnement](#) (int desc, char *params)

3.2.2 Description détaillée

Fonctions de base d'utilisation des sockets.

Auteur:

Mickaël Barroux & Philippe Chen

Version:

1.0

Date:

28 novembre 2008

3.2.3 Documentation des fonctions

```
void Abonnement ( int    desc,  
                 char *  params  
                 )
```

Cette fonction permet d'ajouter un utilisateur à la liste des abonnés d'un sujet

Paramètres:

desc : socket identifiant la connexion

params : login de l'utilisateur et nom du sujet auquel s'abonner

```
void adduser ( int    desc,  
             char *  question
```

)

Cette fonction permet d'ajouter un utilisateur à la liste des utilisateurs du forum. Elle vérifie que le login demandé n'existe pas et ajoute les informations passées en paramètres dans le fichier 'user'. Elle renvoie au client 0 si l'ajout s'est bien déroulé ou 1 sinon.

Paramètres:

desc : descripteur de socket identifiant la connexion

question : chaîne de caractères contenant le login:password:mail de l'utilisateur voulant s'inscrire

```
int Creer_Serveur ( int port,  
                  struct sockaddr_in * Nom_Socket  
                  )
```

Cette fonction crée au niveau TCP une connexion de type serveur

Paramètres:

port : Port que l'on souhaite réserver pour le serveur (0 pour laisser le système choisir)

Valeurs retournées:

Nom_Sock est une structure de type `sockaddr_in` contenant les informations sur le
et socket ouvert par le serveur

Renvoie:

Le descripteur du socket ouvert (-1 si erreur)

```
void EcritureLog ( char * operation )
```

Cette fonction permet d'écrire un message dans le fichier de log du forum

Paramètres:

operation : L'opération à écrire dans le fichier de log

```
int Emettre ( int desc,  
             char * message  
            )
```

Cette fonction permet d'émettre des caractères depuis une connexion identifiée par le descripteur de socket desc

Paramètres:

desc : descripteur de socket identifiant la connexion
message : tableau de caractères à émettre

Renvoie:

-1 si erreur de transmission

```
void Fermer ( int desc )
```

Cette fonction permet la fermeture de la connexion désignée par le descripteur de socket desc

Paramètres:

desc : descripteur de socket identifiant la connexion

```
void lister_messages_sujet ( int desc,  
                           char * nom_sujet  
                          )
```

Cette fonction permet de lister les différents messages du sujet passé en paramètre

Paramètres:

desc : Descripteur du socket sur lequel le client est connecté
nom_sujet : Nom du sujet à lister

```
void ListeTopics ( int desc )
```

Cette fonction permet de récupérer la liste des sujets sur le forum (1 sujet = 1 fichier sur le serveur)

Paramètres:

desc : Descripteur du socket sur lequel le client est connecté

Valeurs retournées:

liste est la liste des sujets créés sur le forum à ce jour

```
void neutraliserEspaces ( char * chaine )
```

Cette fonction permet de remplacer les espaces d'une chaîne de caractères par des underscores

Paramètres:

chaine : chaîne de caractère à neutraliser

```
void NouveauMessage ( int desc,  
                     char * user,  
                     char * topic_name,  
                     char * msg  
                     )
```

Cette fonction permet de créer un nouveau message sur un sujet du forum

Paramètres:

desc : Descripteur du socket sur lequel le client est connecté

user est le nom de l'utilisateur qui poste le nouveau message

topic_name est le titre du sujet sur lequel on poste

msg est le message posté

```
void NouveauSujet ( int    desc,  
                  char * nom_sujet  
                  )
```

Cette fonction permet de créer un nouveau sujet sur le forum

Paramètres:

desc : Descripteur du socket sur lequel le client est connecté
nom_sujet est le titre du sujet créé

```
void oubli_pass ( int    desc,  
                char * question  
                )
```

Cette fonction permet de renvoyer le mot de passe de l'utilisateur

Paramètres:

desc : descripteur de socket identifiant la connexion
question : paramètres de la requête contenant le login et le mail de l'utilisateur ayant perdu son mot de passe

```
void RandomString ( char * str,  
                  size_t n  
                  )
```

Cette fonction permet de générer une chaîne de caractères aléatoirement de taille passée en paramètre.

Paramètres:

n : taille de la chaîne à générer

Valeurs retournées:

str : chaîne de caractères générée aléatoirement

```
int Recevoir ( int desc,  
              char * tampon,  
              int longueur_tampon  
            )
```

Cette fonction permet de recevoir des caractères depuis une connexion identifiée par le descripteur de socket *desc*

Paramètres:

desc : descripteur de socket identifiant la connexion

longueur_tampon : donne la longueur maximum du tampon de réception

Valeurs retournées:

tampon est le tableau de caractères contenant les caractères reçus

Renvoie:

Le nombre de caractères effectivement reçus (-1 si erreur)

```
int recup_id_message_max ( char * nom_sujet )
```

Cette fonction retourne l'id du dernier message posté sur le sujet passé en paramètre

Paramètres:

nom_sujet : Nom du sujet à évaluer

Renvoie:

l'id du dernier message posté sur le sujet

```
void recup_id_topic ( int desc,  
                    int id
```

```
)
```

Cette fonction permet de récupérer le nom du sujet qui porte le numéro id

Paramètres:

desc : Descripteur du socket sur lequel le client est connecté

id : Numéro du sujet (numérotés dans l'ordre alphabétique)

```
int Serveur_Attendre_Client ( int descripteur_socket )
```

Cette fonction attend une connexion sur le descripteur de socket

Paramètres:

descripteur_socket : descripteur de socket renvoyé par Creer_Serveur

Renvoie:

Le descripteur de connexion (-1 si erreur)

```
void SupprimerSujet ( int desc,  
                     char * nom_sujet  
                     )
```

Cette fonction permet de supprimer un sujet sur le forum

Paramètres:

desc : Descripteur du socket sur lequel le client est connecté

nom_sujet est le nom du sujet à supprimer

```
void * thread_client ( void * p_data )
```

Cette fonction gère les requêtes d'un client connecté sur la socket passée en paramètre

Paramètres:

p_data : descripteur de socket sur lequel est connecté le client

```
void VerifConnexion ( int    desc,  
                    char * question  
                    )
```

Cette fonction permet de vérifier si le login et mot de passe passés en paramètres sont corrects pour se connecter au forum. Elle renvoie au client 0 si l'identification est correcte ou 1 sinon.

Paramètres:

desc : descripteur de socket identifiant la connexion

question : chaîne de caractères contenant le login:password de l'utilisateur voulant s'identifier