PROJET C– CLIENT/SERVEUR

Partie Conception

Langage C

POUSSARD Sébastien & BREDARIOL Romain

Table des matières

Introduction

Le But de ce TP était de comprendre les aspects d'un serveur de partage de fichier. Ce TP a été découpé en plusieurs partie afin de nous initier à une bonne organisation de travail. Cette partie traitera donc de la conception globale des fonctions principales du projet. L'ensemble des algorithmes qui vont suivre sont présentés dans un langage de haut niveau. Le client, quant à lui, sera un client console et non web cette fois (cf.TP3). Les algorithmes sont découpés en 2 parties, serveur et client, et s'organisent en 2 niveaux de raffinages.

I. Premier niveau de raffinage

Dans cette partie nous aborderons les fonctions « main » du client et serveur.

1. Côté client

```
Algorithme mainServeur :
/* Role : cree un serveur et communique avec un client distant */

Variables :
    /*message recu du serveur*/
    /*machine est le nom du serveur*/
    message, machine : chaine

Debut :
    /*etablit une connexion avec le serveur*/
    Initialisation(machine)

    /*Tant que le client recoit des messages du serveur*/
    Tant que( (message <- Reception()) != NULL)

        /*Si le client recoit l'ordre de s'identifier*/
        Si(message == "002")
            authentification()
        Fin Si

        /*Si le client recoit la confirmation d'identification
         * d'utilisateur lambda*/
        Si(message == "004")
            affichageMenu(2)
            choix_menu()
        Fin Si

        /*Si le client recoit la confirmation d'identification de
         * super utilisateur*/
        Si(message == "005")
            affichageMenu(3)
            choix_menu()
        Fin Si
    Fin Tant que

    /*ferme la connexion*/
    Terminaison()
Fin
```

2. Côté serveur

```
Algorithme mainServeur :
/* Role : cree un serveur et communique avec un client distant */
Variables :
    fini : entier /*Indicateur du statut de la connexion*/
    requete : chaine /*Requete du Client*/
Debut :
    /*Ouvre le serveur sur un port*/
    initialisation()
    /*boucle infini*/
    tant que(1)
        fini <- 0
        /*Attend la demande de connexion d'un client*/
        Attente()
        /*Si le client s'identifie correctement*/
        si(authentification() != 0)
            /*Tant que la connexion avec le client est active*/
            tant que(fini != 0)
                /*Si le serveur recoit des donnees*/
                requete <- Reception()
                si(requete != 0){
                    /*Execute la requete demande par le client
                     * s'il a les droits*/
                    ExecuterRequete(requete)
                sinon
                    /*Sinon on sort de la boucle pour fermer la
                     * connexion*/
                    fini <- 1
                Fin Si
            Fin Tant que
        sinon
            /*Affiche un message d'erreur*/
            Ecrire("Erreur d'authenfication, veuillez ressayer")
        Fin Si
        /*ferme la connexion*/
        TerminaisonClient()
    Fin Tant que
Fin
```

II. Deuxième niveau de raffinage

Dans ce deuxième niveau de raffinage, nous verrons les fonctions appelé dans la partie précédente plus en détails.

1. Côté client

```
Algorithme Initialisation(machine : chaine) :
/*Role : Etablit une connexion avec le serveur*/

Debut :
    /*la variable machine permet d'identifier le serveur a contacter*/
    /*13214 est le port ouvert par le serveur*/
    InitialisationAvecService(machine, "13214")
Fin

================================================

Algorithme Reception() :
/*Role : recoit un message du serveur*/

Variable :
    tampon : chaine /*chaine recu par le serveur*/
Debut :
    tampon <- ReceptionServeur()

    Si(tampon == NULL)
        Retourne NULL
    Sinon
        Retoune tampon
    Fin Si
Fin

================================================
Algorithme choix_menu() :
/*Role : envoie le choix du menu au serveur*/

Variable :
    choix : entier /*choix du menu*/
Debut :
    Lire(choix)
    Emissin(choix)
Fin

================================================

Algorithme Terminaison() :
/*Role : ferme la connexion avec le serveur*/

Variable :
    socket : entier
Debut :
    close(socket)
Fin
```

```
Algorithme authentification() :
/*Role : permet de s'identifier aupres du serveur*/

Variable :
    login, mdp, donnee : chaine /*identifiant et mot de passe de l'utilisateur*/

Debut :
    Ecrire("Veuillez vous identifier")
    Ecrire("login :")
    Lire(login)
    Ecrire("Password :")
    Lire(mdp)

    /*003 est le code serveur pour l'envoie de donnee d'identification cf.RFC*/
    donnee <- "003 "+login+" "+mdp
    /*envoie les donnees au serveur*/
    Emission(donnee)
Fin

================================================

Algorithme affichageMenu(user : entier) :
/*Role : affiche le menu possible en fonction de l'utilisateur*/

Debut :
    Ecrire("1 - Televerser")
    Ecrire("2 - Telecharger")
    Ecrire("3 - Autorisation users")
    Ecrire("4 - Etat")
    Ecrire("5 - Gerer fichiers")
    Ecrire("6 - Liste fichiers telechargeable")
    /*Si le client est super utilisateur*/
    Si(user == 3)
        Ecrire("7 - Gestion comptes")
    Fin Si
    Ecrire("0 - Quitter")
Fin
================================================
```

2. Côté serveur

```
Algorithme initialisation() :
/*Role : ouvre une connexion sur un port*/

Debut :
    /*ouvre le serveur sur le port 13214*/
    InitialisationAvecService("13214")
Fin

==========================================

Algorithme Attente() :
/*Role : attend qu'un client essaie de se connecter sur le serveur*/

Debut :
    AttenteClient()
Fin

==========================================

Algorithme authentification() :
/*Role : demande au client de s'identifier*/

Variable :
    user : chaine
Debut :
    Emission("001 authentification")
    user <- Reception()

    Si(user_existant())
        Emission("002 auth OK")
        Retourne 1
    Sinon
        Retourne 0
    Fin Si
Fin

==========================================

Algorithme Reception() :
/*Role : recoit un message du serveur*/

Variable :
tampon : chaine /*chaine recu par le client*/

Debut :
    tampon <- ReceptionClient()

    Si(tampon == NULL)
        Retourne NULL
    Sinon
        Retoune tampon
    Fin Si
Fin

==========================================
```

```
Algorithme ExecuterRequete(requete : chaine) :
/*Role : execute la requete demande par le client*/

Variable :

Debut :
    Si(requete == 1)
        televerser()
    Fin Si
    Si(requete == 2)
        telecharger()
    Fin Si
    Si(requete == 3)
        autoriserUser()
    Fin Si
    Si(requete == 4)
        get_Etat()
    Fin Si
    Si(requete == 5)
        gestionFichier()
    Fin Si
    Si(requete == 6)
        get_Fichier()
    Fin Si
    Si(requete == 7)
        gestionCompte()
    Fin Si
    Si(requete == 0)
        quitter()
    Fin Si
Fin

==========================================


Algorithme TerminaisonClient() :
/*Role : ferme la connexion avec le serveur*/

Variable :
    socket : entier

Debut :
    close(socket)
Fin
```