Groupe : Baptiste Bouvier - Damien Wykland - Ancelin Serre

TP5 Réseaux : Programmation d'une application client/serveur

Compte-Rendu

Cahier des charges :

Durant ce TP, nous avons eu pour objectif de réaliser une application client/serveur en utilisant le langage C. Pour ce faire, il nous a été demandé d'illustrer ce programme avec un jeu classique : **le jeu du pendu**. Nous allons donc résumer dans ce cahier des charges les différentes tâches que les deux programmes principaux *client.c* et *serveur.c* auront à réaliser afin de permettre à un utilisateur de jouer convenablement.

Rappel des règles du jeu du pendu :

Le jeu du pendu dispose de règles relativement simple. D'ordinaire, c'est un jeu qui se joue à deux. Dans un premier temps, le joueur 1 se charge de déterminer un mot à faire deviner au joueur 2. Une fois se mot trouvé par le joueur 1, ce dernier se doit de donner le nombre de caractères du mot au joueur 2. Dans un second temps, on entre dans la phase de jeu, dans cette dernière, le joueur 2 à pour mission de trouver le mot choisi par le joueur 1 en donnant un chaque tour une lettre au joueur 1 ; si la lettre donnée est présente dans le mot, le joueur 1 met à jour le mot vu par le joueur 2 avec les lettres trouvées. Dans le cas contraire, le joueur 1 commence à dessiner un pendu avec un trait. Lorsque le mot est trouvé, le joueur 2 remporte la partie, lorsque le pendu est totalement dessiné, le joueur 1 remporte la partie.

Adaptation à notre application :

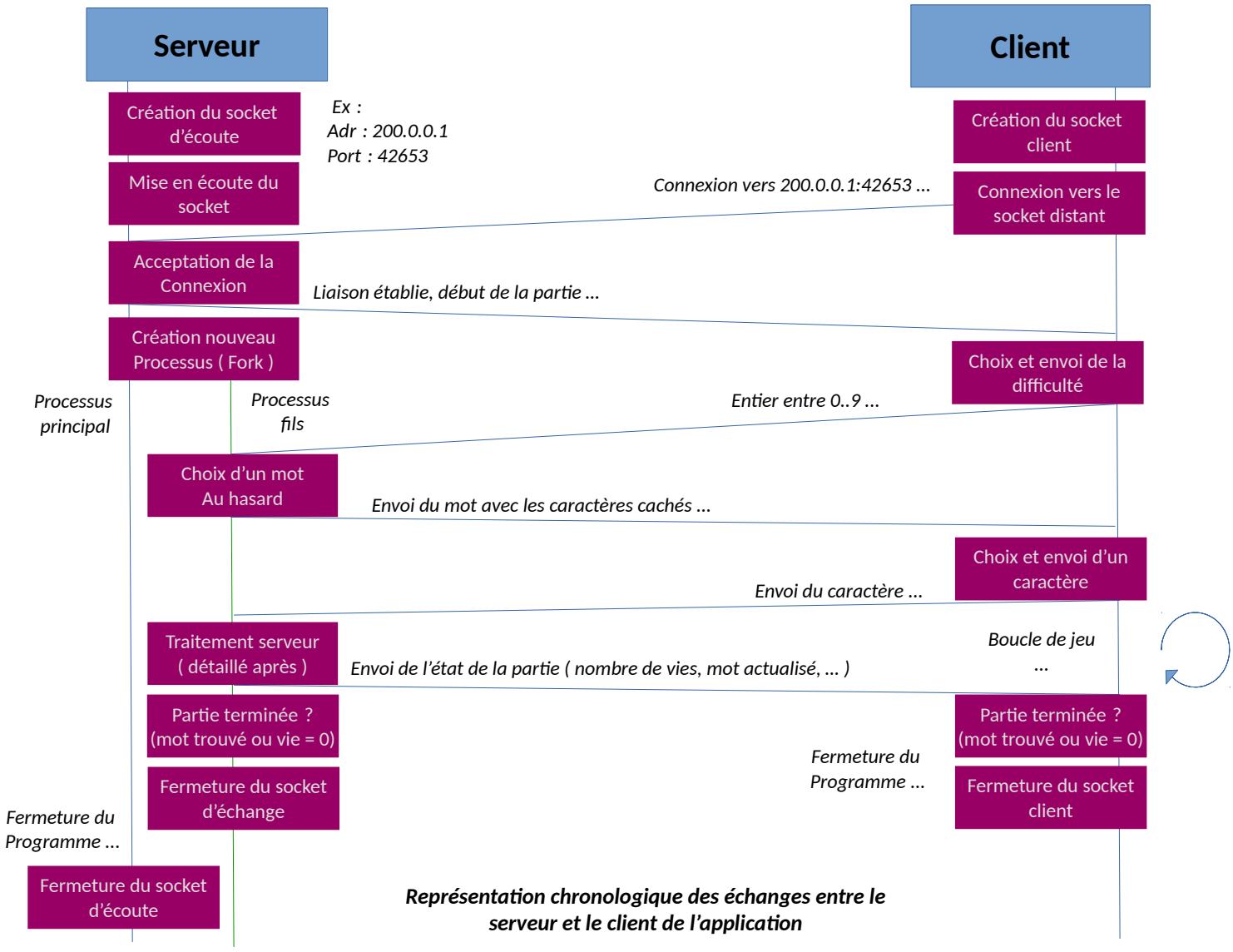
Dans notre cas particulier, il n'y aura qu'un seul et unique joueur : le client. Le serveur aura pour sa part le rôle du maître de jeu qui s'apparente au joueur 1. En effet, il aura pour tâches principales d'initialiser le mot à trouver et de vérifier à chaque tour si la lettre donnée est présente dans ce même mot. Si la lettre est présente, le serveur en informera le client en mettant à jour le mot vu par le ce dernier de la même manière que le joueur 1 le ferait avec le joueur 2 dans le cadre d'une partie classique. La différence principale entre le jeu classique et notre version est que le serveur ne va pas dessiner un pendu sur le terminal du client ce qui en plus d'être laborieux ne serait pas très esthétique, voire visuel. Nous avons donc choisi de faire choisir le nombre de vies au joueur (donc au client) ce qui revient à choisir la difficulté de la partie car il devra choisir ce nombre de vies avant de connaître la longueur du mot. Le serveur se chargera donc de mettre à jour ce nombre de vies si une lettre donnée n'est pas présente dans le mot. Finalement, la partie est remportée par le client lorsque ce dernier trouve le mot (idem jeu classique) ou la partie est perdue par le client lorsque ce dernier n'a plus de vies.

Choix techniques :

- ***Protocole Transport*** : Même si il est souvent d'usage d'utiliser le protocole UDP quand on travaille avec des jeux vidéos, nous avons choisi d'utiliser le protocole TCP pour avoir un serveur orienté « connexion » d'abord parce que dans notre cas, on a un simple dialogue entre le client et le serveur qui suit un parcours bien déterminé donc l'UDP n'aurait pas eu de grand intérêt (le client est toujours à l'initiative de l'échange) et parce qu'il est plus aisé de mettre en place le protocole TCP via les fonctions qui nous sont fournies.

- ***Serveur parallèle*** : Afin de proposer le jeu à plusieurs personnes en même temps, nous avons choisi d'implémenter un serveur parallèle capable de créer des processus différents pour chaque nouvelle partie. Pour cela, nous avons été amené à utiliser la fonction C : *fork()* permettant de créer un processus fils du processus en cours (père).

Algorithme de l'application :



Représentation chronologique des échanges entre le serveur et le client de l'application

Traitement serveur :

Lors de chaque tour de jeu, le **serveur** effectue les tâches suivantes :

1. Réception du caractère envoyé par le client
2. Mise à jour du mot vu par le client
3. Si la lettre n'est pas présente alors le joueur perd une vie
4. Envoi des données mises à jour au client (mot, nombre de vies, état de la partie)

Traitement client :

Lors de chaque tour de jeu, le **client** effectue de son côté les tâches suivantes :

1. Récupération et affichage du mot vu par le client (ex : **BL* si le mot à chercher est TABLE)
2. Demande de choisir un nouveau caractère et envoi de ce dernier au serveur
3. Affichage du résultat de ce choix (Lettre présente dans le mot ou non), affichage du nombre de vies restantes

Autres informations :

- Pour tester notre programme, nous avons utilisé une variable représentant un tableau de mots afin de nous faciliter la tâche. Le mot à trouver est choisi au hasard par le serveur parmi ce tableau. Si on devait améliorer le programme, il serait préférable d'utiliser un fichier dans lequel on pourrait stocké un dictionnaire de mots.

- Dans un soucis pratique, l'affichage est entièrement réalisé par le client. On a fait en sorte de laisser au serveur un travail d'analyse de telle sorte qu'il traite les données que le client lui envoi et renvoi au client uniquement l'essentiel comme un drapeau pour savoir si la partie est terminée ou pas, ou encore, le nombre de vies restantes.

Notice d'utilisation :

- Dans l'archive, **se placer dans le dossier principal** où se trouve le Makefile et exécuter la commande *make*. (si la commande ne fonctionne pas, il est possible que vous deviez installer un paquet. Pour l'installer, veuillez exécuter la commande donnée dans le fichier **lisezmoi**)
- **Configurez votre réseau** comme vous le souhaitez via la commande ifconfig. (vous pouvez par exemple réaliser la configuration suivante : *sudo ifconfig eth0 200.0.0.1/24*)
- Sur un **premier terminal**, lancez le **programme serveur** via la commande *./serveur* (vous pouvez ajouter en argument le numéro de port)
- Sur un **second terminal**, lancez le **programme client** via la commande *./client adr_serv port_serv*
- Suivez les instructions du programme client et jouez comme vous le souhaitez !

N.B. Vous trouverez la totalité du code source dans l'archive ci-jointe.