Rapport
Sherlock 13 : Un jeu en réseau

Par
CHOI Félix
OULD AMER Thizirie

1er juin 2018

Encadrant : M. François Pêcheux.

Table des matières

Introduction

L'homme a toujours aimé jouer, ainsi il créa le jeu. Plus tard, les jeux se sont diversifiés et naquit le jeu de société. Aujourd'hui les gens sont pressés, et trop fatigués pour organiser une soirée jeu de plateau. Mais l'Homme a su communiquer avec le reste du monde depuis un point fixe (chez eux), c'est le réseau internet. Les jeux n'y ont pas échappé, et il est aujourd'hui important pour l'industrie du jeu de société de s'installer dans les jeux en réseau, dans l'internet. Nous allons aujourd'hui retranscrire un jeu de plateau Sherlock 13 en un jeu en réseau. Pour cela il nous faudra exploiter différentes notions. Nous verrons donc ce qu'est le protocole TCP, ainsi que le fonctionnement d'un système serveur-clients, puis nous aborderons les processus parallèle pour terminer avec l'interface graphique.

I Définitions, explications, concepts

I - 1 La communication

L'essence même de ce qu'on a appris durant nos heures passées en cours se trouve dans la communication. Ainsi, les questions qu'on a pu se poser étaient (liste non exhaustive) "Comment ça se passe, lorsque je veux me connecter quelque part ?", "Comment est-ce possible que mon message arrive, intact, à l'autre bout du monde ?", et bien sûr "Jordy et sa copine Léantine à Vancouver, est-ce une histoire qui va durer ?" Pour la dernière, seul le temps nous le dira. Pour les autres, Pêcheux nous a dit.

Le réseau est un moyen de communication (ensemble de machines/applications atteignables). Il existe des modes de communication différents. On peut en citer deux :
- par paquet (datagram) → Pas d'ordre dans la délivrance, risque de perte de données. (ex UDP)
- en flux (stream) → Les informations sont remises dans l'ordre d'émission, pas de perte... On parlera donc ici de TCP qui est un élément principal d'une des couches de TCP/IP.

Transmission Control Protocol - TCP

Le TCP est un protocole de transmission de données fiable, dans la couche transport du modèle TCP/IP. Il découpe le flux d'octets, de données transmises par les applications, en segments.
Son fonctionnement se fait en 3 phases :
- Établissement de la connexion
- Transferts de données
- Fin de la connexion.

Il permet donc, au niveau des applications, de gérer les données en provenance (ou à destination) de la couche inférieure du modèle (c'est-à-dire le protocole IP). L'avantage du TCP c'est qu'il permet à deux machines qui communiquent de contrôler l'état de la transmission de leurs données.

Il permet de vérifier le flot de données afin d'éviter une saturation du réseau, et également de faire circuler simultanément des informations provenant d'applications distinctes sur une même ligne.

I - 2 Un modèle : Un serveur ↔ Des clients

Le serveur

Un serveur est un ordinateur qui s'ouvre aux clients via son adresse et un numéro de port. Il est responsable du fonctionnement du système, notamment en gérant, les messages entrant et sortant. En effet, un serveur tourne en permanence et attend la requête de clients avant d'y répondre. Lorsqu'un client se connecte au serveur, le serveur ouvre un 'socket' qui lui permet de communiquer avec le client. C'est ce qui va, dans notre cadre d'application, gérer et faire tourner le jeu.
/* 1. Socket 2. portno 3. bind 4. listen Boucle : 5. accept 6. read 7. close
*/

Le client

Le client est l'autre partie du système serveur-client, c'est un ordinateur qui se connecte a un serveur en connaissant son adresse ip et le numéro de port par lequel ils vont communiquer. Une fois connecté, le socket étant ouvert depuis le serveur, le client envoie une requête et attend ou non une réponse. C'est la partie joueur si on veut, ou l'interface graphique est gérée
/* 1. portno 2. socket 3. get host 4. connect vs accept 5. write vs read 5. read vs write serveur 6. close */

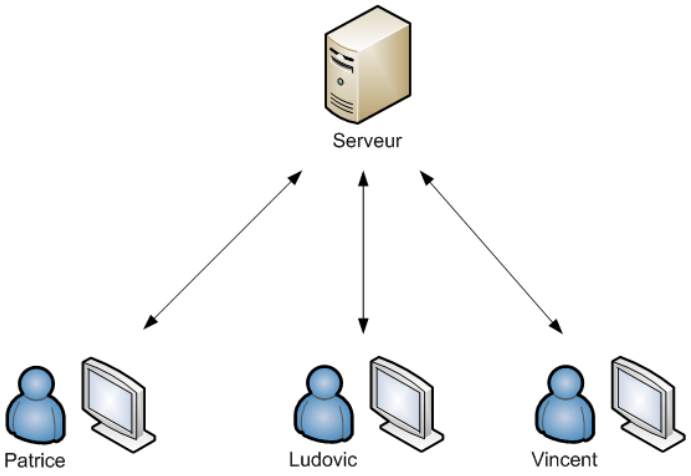


FIGURE 1 – schéma d'un système serveur-client

I - 3 Les appels systèmes

Sockets/bind

Le socket est une sorte de prise, qui permet de créer un point de communication entre un serveur et un client.
Cet appel système renvoie un descripteur de fichier qui sera ensuite nommé avec bind. Il est central dans la programmation réseau puisque sans lui, on ne reçoit rien...
Pour faire simple, imaginons le père Noel qui serait le serveur. Les enfants sont les clients. Pour que l'un laisse un message (ou un cadeau, bien mieux) à l'autre, il faudra laisser ça sous le sapin (on n'a pas trouvé d'analogie pour ce cas) ou mettre ça dans des ... CHAUSSETTES ! Chaussettes=Sockets

listen et accept

Mais socket ne fait pas tout, tout seul ! Chez le serveur, il y a listen, qui lui permet d'attendre que le nombre de connexions limite soit atteint. Une file va donc être crée, ce qui permettra de passer à la phase accept, où ces connexions sont acceptées dans leur ordre d'entrée. Ce sera suivi de connect qui va permettre de connecter le socket associé au descripteur de fichier sockfd.

Threads/forks

Thread, ou fil d'exécution.
DISCLAIMER : les pthreads ne sont pas des appels systèmes, mais des fonctions. Il s'agit d'une fonction qui se lance simultanément avec le thread principal. Ici, ça voudrait dire que plusieurs fonctions vont se lancer en même temps que le processus, et c'est bien ce dont rêve toute personne voulant faire de la programmation multitâche ! C'est à différencier du fork qui, lui, descend du père. C'est un processus (le père) qui sera dupliqué (fils)

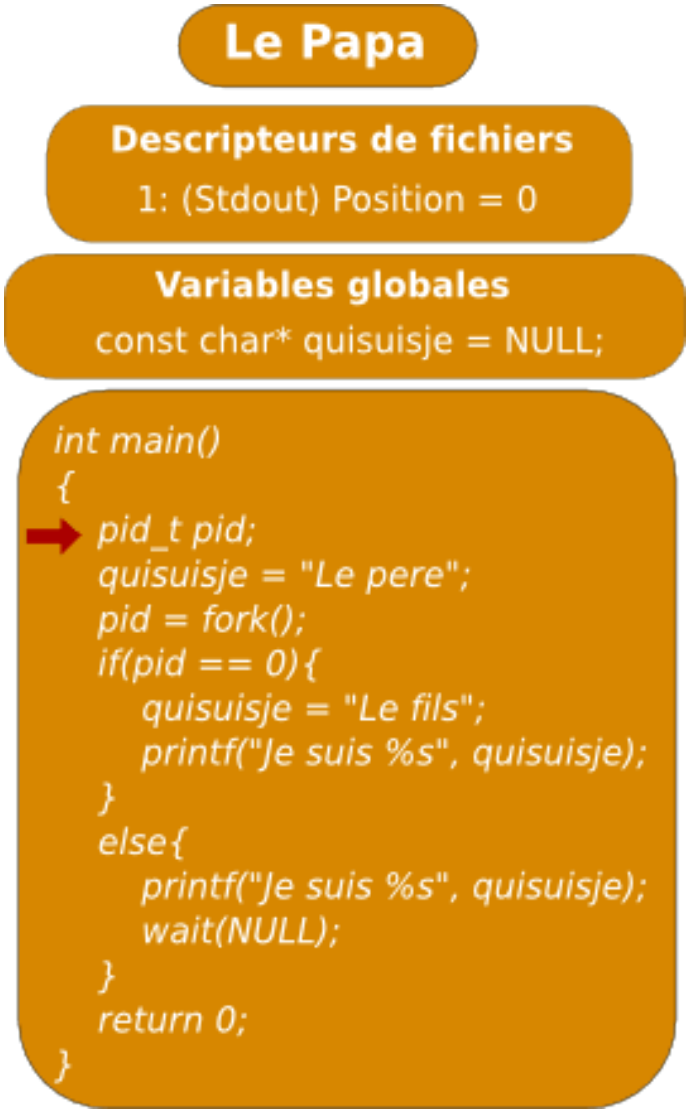


FIGURE 2 – Fonctionnement d'un fork, en C

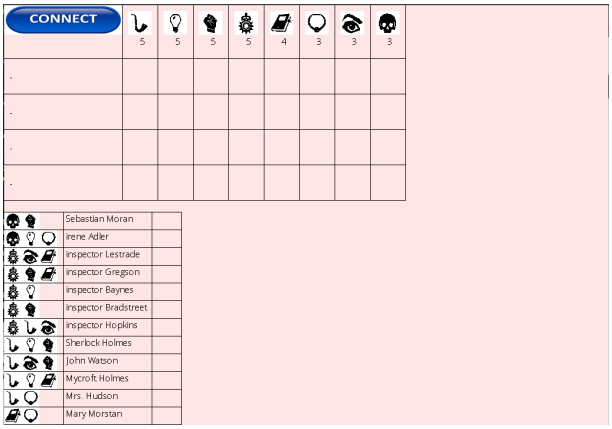
Infos

Ne jamais oublier de fermer le descripteur de fichiers sockfd à la fin de l'utilisation.
Ne pas mettre de fork dans des boucles infinies

II Le jeu

II - 1 Interface graphique

Pour l'interface graphique, on a voulu faire les choses en grand, c'est a dire qu'on tout revisiter, pour donner une petite ambiance. Nous avons réorganisé la disposition du plateau ainsi que le design général. Sur le code donné, il a donc fallut en réécrire environ 70%.



⇓

II - 2 Tutoriel pour le jeu

Jeu à 4 joueurs.

1. Chacun doit se connecter sur son terminal après le lancement du serveur.
2. En lançant le serveur il faut spécifier le numéro de port qu'il va prendre. Sachant que pour l'adresse IP, il prendra celle de la machine sur laquelle il est lancé.
3. Chaque joueur devra, par contre, spécifier 5 champs. Le port du serveur, suivie de son adresse IP. Ensuite le numéro de port du joueur et son IP à lui. Puis enfin son identifiant.
4. Le premier à s'identifier sera le joueur 0 ... etc.
5. Une fenetre s'affichera pour chaque joueur, et ils devront tous se connecter.
6. Chaque joueur aura 3 cartes sur les 13 possibles. Le but et de trouver la seule qui n'a pas été distribuée = le coupable. Chaque carte a des éléments "caractéristiques".
7. Le joueur 0 va commencer à jouer. et ainsi de suite.
8. Trois choix s'offrent à chacun.
Demander combien de "caractéristique X" a la personne Z.
Demander si tous les autres joueurs ont le caractéristique X.
Proposer un coupable.
9. Le gagnant sera celui qui trouvera le coupable en premier.
Happy Sherlock games.

Conclusion

Après avoir appris énormément de choses, sur les appels systèmes, la communication et la programmation réseau (et même sur le langage C), nous sommes arrivé.e.s à une conclusion évidente. Après 2 mois de travail, nous avons la réponse. Le cours nous a beaucoup aidé à comprendre beaucoup de nouvelles notions, et les TPs les ont consolidées. Et c'est grâce à ça que la solution est entre nos mains aujourd'hui. (en plus du rapport et du programme qui marche). La solution est :

42.

Bibliographie

https ://fr.wikipedia.org/wiki/Mod%C3%A8le_OSI
https ://fr.wikipedia.org/wiki/Van_Jacobson
https ://fr.wikipedia.org/wiki/Suite_des_protocoles_Internet
https ://www.web24.com.au/library/getting-started-servers-everything-need-know
https ://www.lifewire.com/servers-in-computer-networking-817380
https ://franckh.developpez.com/tutoriels/posix/pthreads/ https ://man.