Projet OS : Sherlock 13

Liza Idri

April 2025

1 Introduction

Ce projet a pour objectif de développer le jeu "Sherlock 13", réparti en deux programmes :

Le serveur (server.c) : Gère les connexions TCP, la distribution des cartes, et le déroulement des tours.

Le client SDL (sh13.c) : Gère l'interface graphique, l'envoie des requêtes au serveur et affiche l'état du jeu.

2 Fonctionnement général

2.1 Communication serveur-client

2.1.1 Messages du client vers serveur

"C" : C <ipClient> <portClient> <nom>
Le client envoie une requête de connexion lorsque l'utilisateur clique sur "Connect". Traitement : Le serveur, en mode fsmServer==0, ajoute ce client à la liste tcpClients[], incrémente nbClients, et lui répond immédiatement par un message "I <id>" suivi d'un broadcast "L <noms des 4 joueurs>" une fois les quatre joueurs connectés.

G = G <idClient> <carte>
Lorsqu'un joueur clique sur le bouton "Go" en mode accusation, le client envoie "G id guiltSel".
Le serveur, en mode fsmServer==1, lit cette ligne avec sscanf, avance le joueur courant, et broadcast "M nouveauJoueur" pour signaler le tour suivant.

O = O <idClient> <objet>
Pour poser une question sur un objet (sans viser de joueur particulier), le client envoie "O id objetSel". Le serveur parcourt tous les joueurs i, et renvoie au demandeur autant de messages "V i ¡objet¿ ¡valeurTable¿" que nécessaire. Ensuite, il met à jour joueurCourant et effectue un broadcast "M nouveauJoueur".

S = S <idClient> <idCible> <objet>
En mode suggestion (objet + joueur cible), le client envoie S id joueurSel objetSel. Le serveur répond par un unique message V ¡joueurSel¿ ¡objet¿ ¡valeur¿ réservé au demandeur, puis passe le tour et broadcast M ¡nouveauJoueur¿.

2.1.2 Messages du client vers serveur

2.2 Phases de jeu

Démarrage du serveur : Création d'une socket TCP (socket, bind, listen), puis boucle accept().

Phase de connexion (fsmServer == 0) : Réception de 4 messages "C"; enregistrement des clients dans tcpClients[], envoi d'un ID (I) et de la liste globale (L).

Distribution : Dès que 4 joueurs sont connectés, mélange du deck, appel à createTable(), puis pour chaque joueur : envoi de ses 3 cartes (D) et de sa ligne de grille (V x y valeur) ; enfin, broadcast du joueur courant initial (M 0).

Phase de jeu (fsmServer == 1) : à chaque réception d'une requête :
- Guess (G id carte) : accusation, on passe simplement au tour suivant et on notifie (M nouveau_id).

- Objet (O id objet) : le serveur renvoie au demandeur la valeur de tableCartes pour tous les joueurs sur cet objet, puis passe au tour suivant (M).
- Suggestion (S id cible objet) : envoie au demandeur la seule valeur correspondant au joueur ciblé et à l'objet, puis notification de tour.

Client : démarre un thread (pthread_create) pour l'écoute TCP locale. Le thread établit une socket, fait bind, listen, accept et stocke chaque message reçu dans le buffer global gbuffer en passant un drapeau synchro=1.

Boucle graphique : traitement SDL des clics : connexion (bouton "Connect"), sélection de joueur/objet/suspect, et clic sur le bouton "GO" pour envoyer G, O ou S. À chaque itération, si synchro==1, on lit gbuffer, on sscanf le code de message.

3 Completion du code

3.1 serveur.c

Distribution des cartes :
- Boucle for (i=0;i¡4;i++) : envoi des 3 cartes "D" par sprintf+sendMessageToClient.
- Double boucle for (j=0;j¡8;j++) : envoi de chaque valeur de grille (V i j tableCartes[i][j]).
- Broadcast du message "M joueurCourant" pour que tous sachent qui commence.

Case 'G' (accusation) : Pour avancer le tour et notifie la nouvelle valeur de joueurCourant.

Case 'O' (question objet) : Retourner au demandeur la valeur de chaque joueur pour l'objet interrogé, puis passe au tour suivant.

Case 'S' (suggestion) : Envoi sélectif de la valeur pour le joueur ciblé et l'objet.

3.2 sh13.c

Bouton Connect : Envoi du message de connexion "C", puis désactivation de l'UI pour éviter les doubles.

Bouton Go : Pour chaque cas, on formate sendBuffer puis on appelle sendMessageToServer().
- Accusation : "G gId guiltSel" si suspicion d'un coupable.
- Question objet : "O gId objetSel" si seul objet sélectionné.
- Suggestion : "S gId joueurSel objetSel" si joueur+objet.

Réception des messages :
case I : lecture de l'ID avec sscanf et stockage dans gId.
Case L : mise à jour du tableau gNames.
Case D : stockage des cartes dans b[0..2].
Case M : activation de goEnabled si cur == gId.
Case V : tableCartes[pid][obj] = val.