

OS User Sherlock13

EISE4

Yining BAO

Sommaire

1. Capture des écrans

2. Réseaux TCP

3. Thread

1. Capture des écrans

```
kzel@kzel-PC: ~/Desktop/code$ ./server 1024
```

```
0 Sebastian Moran
1 irene Adler
2 inspector Lestrade
3 inspector Gregson
4 inspector Baynes
5 inspector Bradstreet
6 inspector Hopkins
7 Sherlock Holmes
8 John Watson
9 Mycroft Holmes
10 Mrs. Hudson
11 Mary Morstan
12 James Moriarty
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
1 irene Adler
3 inspector Gregson
6 inspector Hopkins
10 Mrs. Hudson
2 inspector Lestrade
11 Mary Morstan
0 Sebastian Moran
8 John Watson
7 Sherlock Holmes
5 inspector Bradstreet
9 Mycroft Holmes
4 inspector Baynes
12 James Moriarty
01 01 01 02 01 01 01 01
01 00 00 01 02 02 01 00
02 01 03 00 00 00 01 01
01 02 01 02 01 00 00 00
```

```
Received packet from 127.0.0.1:57070
Data: [C localhost 1025 A
]
```

```
COM=C ipAddress=localhost port=1025 name=A
0: localhost 01025 A
id=0
```

```
Received packet from 127.0.0.1:57076
Data: [C localhost 1026 B
]
```

```
COM=C ipAddress=localhost port=1026 name=B
0: localhost 01025 A
1: localhost 01026 B
id=1
```

```
Received packet from 127.0.0.1:57084
Data: [C localhost 1027 C
]
```

```
COM=C ipAddress=localhost port=1027 name=C
0: localhost 01025 A
1: localhost 01026 B
2: localhost 01027 C
id=2
```

```
Received packet from 127.0.0.1:57094
Data: [C localhost 1028 D
]
```

```
COM=C ipAddress=localhost port=1028 name=D
0: localhost 01025 A
1: localhost 01026 B
2: localhost 01027 C
3: localhost 01028 D
id=3
```

Capture

Figure 1 et 2 : Lancement du serveur avec port de **1024**, le port dynamique est de **1024 à 65535** avec l'adresse de IP **localhost (127.0.0.1)** et réception des adresse IP, le port et le nom des utilisateurs.

```
kzel@kzel-PC: ~/Desktop/code$ ./sh13 localhost 1024 localhost 1025 A
```

```
Sans=0x5591e1de4450
```

```
Creation du thread serveur tcp !
```

```
consomme |I 0
```

```
|
consomme |L A - -
```

```
|
consomme |L A B - -
```

```
|
consomme |L A B C -
```

```
|
consomme |L A B C D
```

```
|
consomme |D 1 3 6
```

```
|
consomme |V 0 0 1
```

```
|
```

```
kzel@kzel-PC: ~/Desktop/code$ ./sh13 localhost 1024 localhost 1026 B
```

```
Sans=0x555be0ff0c70
```

```
Creation du thread serveur tcp !
```

```
consomme |I 1
```

```
|
consomme |L A B - -
```

```
|
consomme |L A B C -
```

```
|
consomme |L A B C D
```

```
|
consomme |D 10 2 11
```

```
|
consomme |V 1 0 1
```

```
|
V 1 0 1
```

```
V Joueur[1] = Carte[0] : 1
```

```
consomme |V 1 1 0
```

```
|
V 1 1 0
```

```
V Joueur[1] = Carte[1] : 0
```

```
consomme |V 1 2 0
```

```
kzel@kzel-PC:~/Desktop/code$ ./sh13 localhost 1024 localhost 1027 C
Sans=0x55c2a1327590
Creation du thread serveur tcp !
consomme |I 2
|
consomme |L A B C -
|
consomme |L A B C D
|
consomme |D 0 8 7
|
consomme |V 2 0 2
|
V 2 0 2
V Joueur[2] = Carte[0] : 2
consomme |V 2 1 1
|
V 2 1 1

kzel@kzel-PC:~/Desktop/code$ ./sh13 localhost 1024 localhost 1028 D
Sans=0x56446a0ced30
Creation du thread serveur tcp !
consomme |I 3
|
consomme |L A B C D
|
consomme |D 5 9 4
|
consomme |V 3 0 1
|
V 3 0 1
V Joueur[3] = Carte[0] : 1
consomme |V 3 1 2
|
V 3 1 2
V Joueur[3] = Carte[1] : 2
consomme |V 3 2 1
|
V 3 2 1
```

Figure 3,4,5 et 6: Lancement des clients, on a besoin de saisir l'adresse IP et le port du serveur qu'on connecte et aussi l'adresse IP, le port et le nom du client.

Après cliquer sur le bouton “connect” le jeu est lancé.

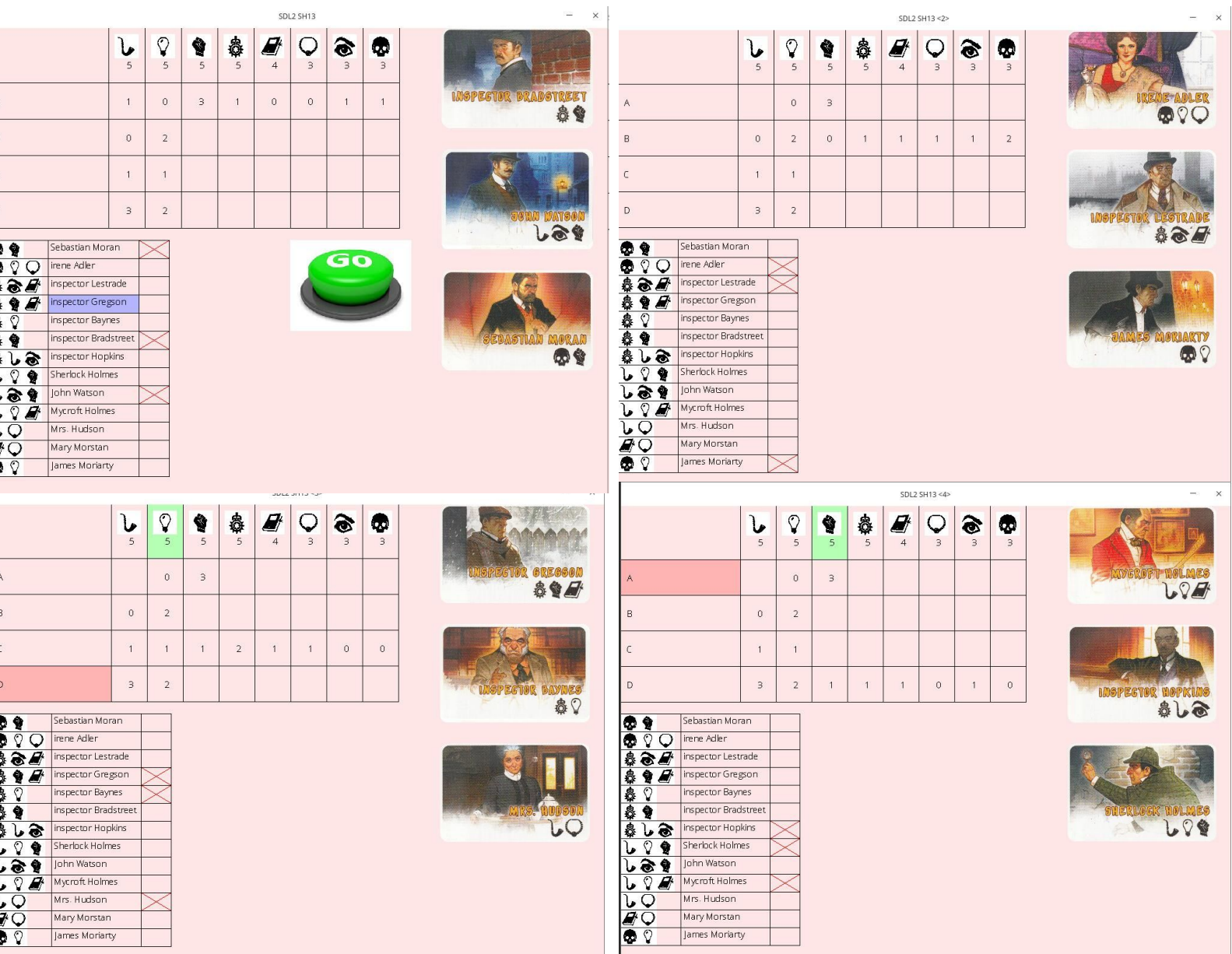


Figure 7,8,9,10: Les fenêtres du jeu

Le bouton “GO” est tourné lorsque chaque tour d’un client à jouer. Après le lancement du jeu, on **distribue aléatoirement** chaque joueur **3 cartes** parmi **13 cartes**, il reste une carte cachée à trouver par des joueurs. Chaque carte est composée des objets par ex : **livre, œil, lampe etc.**

Le joueur peut demander que les autres joueurs maintiennent combien d'objets et devine leurs cartes à la main et la carte cachée à trouver. Pour réaliser cette action, on doit cliquer sur le nom du joueur qu’on veut demander et l’objet et le bouton “GO”.

Pour confirmer **la carte cachée** , on clique sur le nom de la carte et le bouton “GO”.

Lorsqu'on trouve **la carte cachée**, le terminal affiche le résultat et le gagnant.



```
kzel@kzel kzel@kzel kzel@kzel kzel@kzel kzel@kzel + ≡ - □ ×
consomme |V 0 1 0
|
V 0 1 0
V Joueur[0] = Carte[1] : 0
consomme |M 2
|
consomme |V 3 1 2
|
V 3 1 2
V Joueur[3] = Carte[1] : 2
consomme |M 3
|
go! joueur=0 objet=2 guilt=-1
consomme |V 0 2 3
|
V 0 2 3
V Joueur[0] = Carte[2] : 3
consomme |M 0
|
consomme |
Le resultat est: Mary Morstan
Le gagnant est:A
|
consomme |M -1
|
kzel@kzel-PC: ~/Desktop/code$
```

Figure 11: le résultat et le gagnant

2. Réseaux TCP

On peut résumer **TCP** en C par le schéma suivant

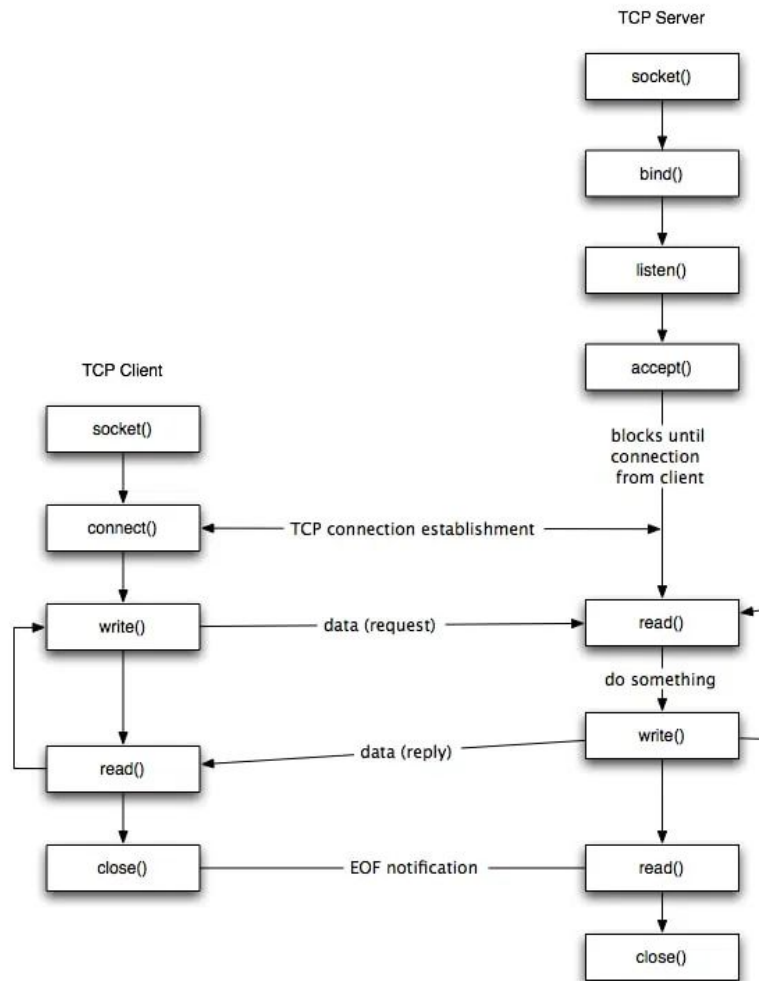


Figure12: Socket programming in c using TCP/IP

Partie Serveur:

Le code ajoute du **serveur** est dans le partie `write()` ou `sendMessageToClient()` avec reply.

On envoie les cartes aux 4 joueurs avec le message "D" pour la réception de 3 cartes de chaque joueur.

Puis le message "V" contient l'indice et le nombre d'objets des cartes que le joueur maintient.

Après le message "D" et "V", on envoie un message "M" à tout le monde pour définir le numéro du joueur courant.

Alors la variable **fsmserver** peut observer l'état du jeu, lorsqu'il égale 0 le jeu est en cours de joindre et le jeu commence quand il égale 1.

Dans la partie suivante `if (fsmServer==1)`

On aura 3 messages "G" "O" et "S" a recevoir par le client.

Le message "G" est que le joueur devine la carte cachée s'il devine correctement on envoie le message du résultat et il gagne, sinon on l'élimine dans le tableau de Joueur Sortie et on passe au joueur suivant.

Si tous les joueurs sont sortis, on envoie un message "M" pour dire que le jeu est fini.
Si le joueur demande une enquête a un joueur, on reçoit le message "S" et on envoie le message "V" qui contient l'indice et le nombres d'objets
Si le joueur demande une enquête a tout le monde, on reçoit le message "O" et on envoie le message "V" qui contient l'indice et le nombre d'objets.

Partie Client:

Dans la partie cliente, on envoie premièrement le message "C" qui contient l'adresse IP, le port et le nom du joueur.

Lorsque le serveur reçoit le message "C", il envoie un message "I" au client pour lui distribuer un ID unique, on peut déterminer le gagnant/perdant avec ID puis il envoie le message "L" a tout le monde.

Le message "L" contient les message le nom des joueurs et on les affiche dans la fenêtre.

Le message "D" "M" et "L", on peut les trouver dans la partie du serveur.

3.Thread

Il y a 2 threads qui sont implementes dans la partie cliente.

`pthread_t thread_serveur_tcp_id;`

`pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;`

Le thread `thread_serveur_tcp_id` permet de créer le thread pour serveur tcp et l'autre thread pour dessiner les images par SDL.

On veut que les 2 threads fonctionnent en même temps et ils ne perturbent pas entre eux, on utiliser une variable volatile `synchro`.

Si le synchro égale 1, le joueur est synchronisé avec le serveur, on peut lancer le jeu et on le verrouille le mutex, après l'action des joueurs avec les cartes, on le mis a 0 et on déverrouille le mutex.