

du jeu "Sherlock 13"

WEI Yongtao

Dans ce projet, nous avons utilisé le protocole TCP (socket), le thread et la bibliothèque graphique SDL2 pour réaliser le jeu "Sherlock 13". Le protocole TCP peut assurer la stabilité de la transmission des données, le thread est utilisé pour synchroniser le client et le serveur, et la bibliothèque graphique SDL2 peut réaliser l'interface visuelle du jeu, ce qui améliore grandement la jouabilité et l'intérêt du jeu.

Sherlock Holmes, John Watson, le professeur Moriarty, Irène Adler ... des personnages bien connus sont devenus des suspects dans une série de crimes. L'un d'eux doit être un criminel! Qui peut trouver le criminel en premier?

2.2 Préparation du Jeu

- ❖ Choisissez un premier joueur Le joueur mélange 13 cartes de personnage, en choisit une au hasard et la pose face cachée sur la table sans voir le contenu de la carte. Le rôle de cette carte est le criminel!
- ❖ Distribuez les 12 cartes de personnage restantes à tous les joueurs: 3 cartes par personne.
- ❖ Chaque joueur prend un déflecteur et un papier d'indice. Utilisez un déflecteur pour bloquer le papier d'indices afin que les autres joueurs ne le voient pas.
- ❖ Avant le début du jeu, chaque joueur enregistre les symboles qui apparaissent sur ses cartes sur le papier d'indices.

2.3 Progression du jeu

Le jeu commence avec le premier joueur et se déroule dans le sens des aiguilles d'une montre. Chaque joueur recueille des informations pour déterminer qui est le criminel. Il existe au total 2 méthodes d'investigation (①, ②). Une fois l'enquête terminée, c'est au tour du joueur suivant de faire son tour. Dans le même temps, le joueur peut également choisir d'accuser le criminel (③). Quand c'est au tour du joueur, le joueur ne peut effectuer qu'une seule action! (①, ② ou ③)

① Enquête 1: Choisissez un symbole et demandez aux autres joueurs qui ont ce symbole. Les joueurs avec ce symbole doivent lever la main. Les joueurs qui posent des questions n'ont pas besoin de lever la main.

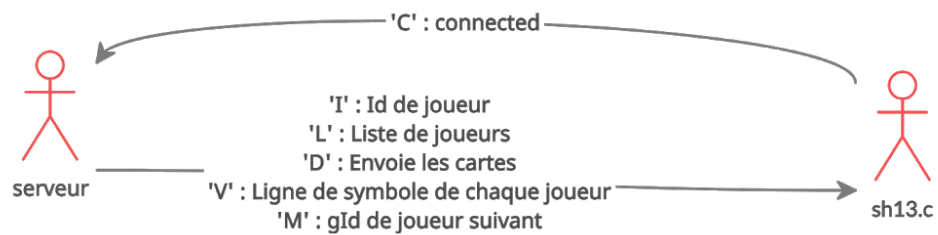
② Enquête 2: choisissez un autre joueur et demandez-lui combien de symboles spécifiques il possède. Le joueur interrogé doit lui dire combien il y a de symboles de ce type.

③ Accusation: dites le nom du criminel que vous pensez être le criminel, puis vérifiez secrètement la carte criminelle sur la table. Seul le joueur qui a initié l'accusation peut voir la carte criminelle. S'il a raison, ouvrez cette carte criminelle et il gagne la partie. S'il se trompe, remettez secrètement la carte criminelle face cachée dans sa position d'origine. Ensuite, pour le reste de la partie, son tour sera sauté, mais il doit encore répondre aux questions des autres joueurs. Une fois l'accusation erronée, le joueur suivant passe à son tour.

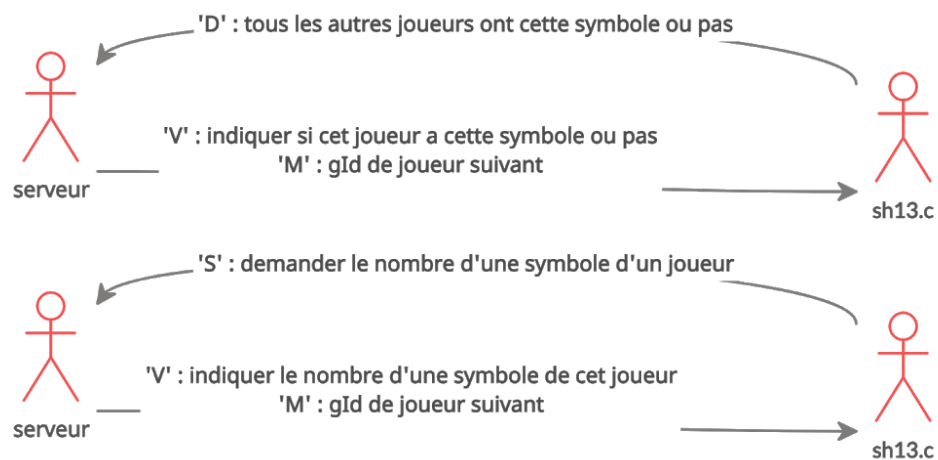
Quand l'accusation d'un joueur est correcte, il devient le gagnant du jeu et le jeu se termine. Ou quand il ne reste qu'un seul joueur qui ne fait aucune fausse accusation, ce joueur devient le gagnant du jeu et le jeu se termine.

3. UML

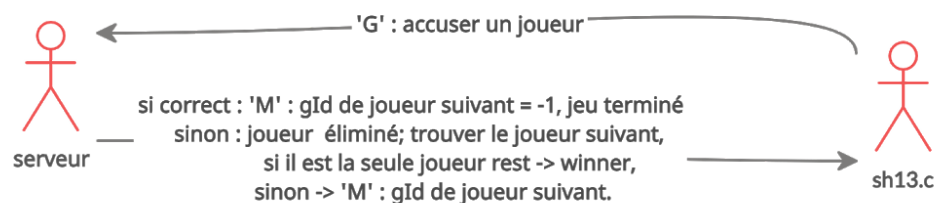
I. CONNECTION



II. PROGRESSION



III. ACCUSATION



4. server.c

4.1. Structure Principale

Tout d'abord, nous utilisons des fonctions pour exécuter les composants nécessaires du jeu: mélanger les cartes; créer une table pouvant être jouée par quatre joueurs; afficher la table; afficher les informations des joueurs et rechercher des joueurs par nom.

- `void melangerDeck()`
- `void createTable()`
- `void printDeck()`
- `void printClients()`
- `int findClientByName(char *name)`

Ensuite, nous utilisons des fonctions pour communiquer avec le client et diffuser les informations à tous les joueurs.

- `void sendMessageToClient(char *clientip,int clientport,char *mess)`
- `void broadcastMessage(char *mess)`

De nombreuses variables importantes sont stockées dans ce programme. Nous utilisons la variable `fsmServer` pour déterminer s'il y a suffisamment de joueurs et nous utilisons `tableCartes [4] [8]` pour stocker les informations de carte des quatre joueurs. La fonction principale répond au joueur en fonction des informations d'en-tête du message envoyé par le joueur et maintient la progression du jeu.

4.2. Code Ajouté

Lorsque les quatre joueurs entrent dans le jeu, nous leur distribuons trois cartes et le nombre de 8 types de symboles.

Pour ce faire, nous utilisons une boucle `for`. Dans chaque boucle, nous envoyons d'abord trois cartes avec la lettre "D" qui pour distinguer les messages, puis nous utilisons une boucle `for` pour envoyer le contenu de `tableCartes` correspondant au joueur avec la lettre "V".

```
for( i=0; i<4; i++){
    // Envoie les cartes
    sprintf(reply,"D %d %d %d",deck[i*3], deck[i*3+1], deck[i*3+2]);
    sendMessageToClient(tcpClients[i].ipAddress, tcpClients[i].port, reply);
    //Envoie la ligne
    for(int j=0; j<8; j++){
        sprintf(reply,"V %d %d %d", i, j, tableCartes[i][j]);
        sendMessageToClient(tcpClients[i].ipAddress, tcpClients[i].port, reply);
    }
}
```

Case G:

Lorsqu'un joueur fait une supposition, si le joueur devine correctement, il devient directement le gagnant et diffuse les informations sur le gagnant et les informations sur les méchants à tous les joueurs; s'il fait une mauvaise estimation, le joueur est éliminé. S'il y a déjà trois joueurs éliminés, le dernier joueur devient automatiquement le gagnant, sinon le

jeu continue.

```
case 'G':
    // RAJOUTER DU CODE ICI
    sscanf(buffer, "G %d %d", &gId, &guiltSel);
    if(deck[12]==guiltSel)
    {
        sprintf(reply, "\nThe gagnant selected is correct: %s\nThe winner is:%s\n",
            nomcartes[guiltSel], tcpClients[gId].name);
        broadcastMessage(reply);
        sprintf(reply, "M %d", -1);
        broadcastMessage(reply);
        fsmServer=0;
    }
    else
    {
        sprintf(reply, "\nEn raison de mauvaise accusation\nLe joueur %s a été éliminé \n",
            tcpClients[gId].name);
        broadcastMessage(reply);
        out_player[gId] = 1;
        int j = 0;
        for (int i = 0; i < 4; i++)
        {
            if(out_player[i]==1)
            {
                j++;
            }
        }
        if (j==3)
        {
            for (int i = 0; i < 4; i++)
            {
                if(out_player[i]==-1)
                {
                    gId = i;
                }
            }
            sprintf(reply, "\nThe gagnant is : %s\nThe winner is:%s\n",
                nomcartes[deck[12]], tcpClients[gId].name);
            broadcastMessage(reply);
            break;
        }
        else{
            while(1){
                joueurCourant = (++joueurCourant)%4;
                if(out_player[joueurCourant]!=1){
                    sprintf(reply, "M %d", joueurCourant);
                    broadcastMessage(reply);
                    break;
                }
            }
        }
    }
}

break;
```

Case O:

Si le message envoyé par le joueur commence par O, cela signifie qu'il choisit un symbole et demande aux autres joueurs qui ont ce symbole. (Correspondant à 2.3.①)

Afin de traiter cette requête, nous utilisons une boucle for pour parcourir tous les joueurs. À l'intérieur de la boucle, si le joueur a ce symbole, le serveur diffuse "V", l'identifiant du joueur, ce symbole et le drapeau "100", sinon, le serveur diffuse "V", l'identifiant du joueur, ce symbole et le drapeau "0".

```

case 'O':
    // RAJOUTER DU CODE ICI
    sscanf(buffer,"O %d %d", &gId, &objetSel);
    for(int i=0;i<nbClients;i++){
        if(tableCartes[i][objetSel]!=0){
            sprintf(reply,"V %d %d %d", i, objetSel, 100);
            broadcastMessage(reply);
        }
        else{
            sprintf(reply,"V %d %d %d",i, objetSel, 0);
            broadcastMessage(reply);
        }
    }
    while(1){
        joueurCourant = (++joueurCourant)%4;
        if(out_player[joueurCourant]!=1){
            sprintf(reply,"M %d",joueurCourant);
            broadcastMessage(reply);
            break;
        }
    }
    break;

```

Case S:

Un joueur demande à un joueur la quantité totale d'un certain symbole (Correspondant à 2.3.②) Pour le répondre, le serveur diffuse cette information à tous les joueurs.

```

case 'S':
    // RAJOUTER DU CODE ICI
    sscanf(buffer,"S %d %d %d", &gId, &joueurSel,&objetSel);
    sprintf(reply,"V %d %d %d", joueurSel, objetSel, tableCartes[joueurSel][objetSel]);
    broadcastMessage(reply);
    while(1){
        joueurCourant = (++joueurCourant)%4;
        if(out_player[joueurCourant]!=1){
            sprintf(reply,"M %d",joueurCourant);
            broadcastMessage(reply);
            break;
        }
    }
    break;

```

5. sh13.c

5.1. Structure Principale

Ce programme établit une connexion TCP via une fonction et communique avec le client via une fonction.

- `void *fn_serveur_tcp(void *arg)`
- `void sendMessageToServer(char *ipAddress, int portno, char *mess)`

Afin d'assurer la bonne connexion entre le client et le serveur, nous ajoutons la variable synchro (volatile). Lorsqu'il est égal à 1, cela signifie que les deux sont correctement connectés et que la synchronisation est maintenue. Afin d'empêcher différents threads d'accéder à cette variable en même temps, nous adoptons une méthode mutuellement exclusive.

- `pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;`
- `pthread_mutex_lock(&mutex);`
- `pthread_mutex_unlock(&mutex);`

5.2. Code Ajouté

Protéger les variables partagées

```
pthread_mutex_lock( &mutex );
synchro=1;
pthread_mutex_unlock( &mutex );
```

Envoyez des informations au serveur. Lorsqu'un joueur appuie sur le bouton «connecter», le processus client du joueur stocke les informations «C» dans le cache, et envoie l'adresse personnelle, le numéro de port et les informations de cache au serveur.

```
if ((mx<200) && (my<50) && (connectEnabled==1))
{
    sprintf(sendBuffer,"C %s %d %s",gClientIpAddress,gClientPort,gName);

    // RAJOUTER DU CODE ICI
    sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
}
```

Le joueur envoie différentes informations au client

```
printf("go! joueur=%d objet=%d guilt=%d\n",joueurSel, objetSel, guiltSel);
if (guiltSel!=-1)
{
    sprintf(sendBuffer,"G %d %d",gId, guiltSel);

    // RAJOUTER DU CODE ICI
    sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
}
else if ((objetSel!=-1) && (joueurSel==1))
{
    sprintf(sendBuffer,"O %d %d",gId, objetSel);

    // RAJOUTER DU CODE ICI
    sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
}
else if ((objetSel!=-1) && (joueurSel!=1))
{
    sprintf(sendBuffer,"S %d %d %d",gId, joueurSel,objetSel);

    // RAJOUTER DU CODE ICI
    sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
}
```

Case I,L,D :

Le message I indique que le joueur a reçu son identifiant de jeu; Le message L indique que le joueur a reçu les informations de la liste des

joueurs; Le message D indique que le joueur a reçu trois cartes

```
case 'I':
    // RAJOUTER DU CODE ICI
    sscanf(gbuffer, "I %d", &gId);
    printf("Mon Id set %d\n", gId);
    break;
// Message 'L' : le joueur reçoit la liste des joueurs
case 'L':
    // RAJOUTER DU CODE ICI
    sscanf(gbuffer, "L %s %s %s %s", gNames[0], gNames[1], gNames[2], gNames[3]);
    printf("la list de noms sont: %s %s %s %s\n", gNames[0], gNames[1], gNames[2], gNames[3]);
    break;
// Message 'D' : le joueur reçoit ses trois cartes
case 'D':
    // RAJOUTER DU CODE ICI
    sscanf(gbuffer, "D %d %d %d", &b[0], &b[1], &b[2]);
    printf("Mon cartes sont: %d %d %d\n", b[0], b[1], b[2]);
    break;
```

Case M:

Le message M indique qu'il faut appuyer sur le bouton "go" pour faire avancer le jeu. Nous déterminons si le joueur a actuellement besoin d'effectuer l'opération, c'est nous-mêmes, et si c'est nous-mêmes, nous activons le bouton «go».

```
case 'M':
    // RAJOUTER DU CODE ICI
    sscanf(gbuffer, "M %d", &joueurCourant);
    printf("joueurCourant est: %d\n", joueurCourant);
    if(joueurCourant == gId)
    {
        goEnabled = 1;
    }
    else
    {
        if(joueurCourant == -1)
        {
            quit = 1;
        }
        goEnabled = 0;
    }

    break;
```

Case V:

Le message V indique que le joueur reçoit une valeur de table des cartes.

```
case 'V':
    // RAJOUTER DU CODE ICI
    sscanf(gbuffer, "V %d %d %d", &player, &objet, &number);
    tableCartes[player][objet] = number;
    printf("player %d a %d objet %d\n",
        player, tableCartes[player][objet], objet);
    break;
```


6. Exécutez le Programme

6.1 Préparation du Jeu

Nous ouvrons le serveur et attendons que les joueurs se joignent.

```
wang@ubuntu:~/s7/os/sh13_etu$ ./server 3200
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
8 John Watson
0 Sebastian Moran
1 irene Adler
10 Mrs. Hudson
12 James Moriarty
6 inspector Hopkins
2 inspector Lestrade
5 inspector Bradstreet
7 Sherlock Holmes
3 inspector Gregson
11 Mary Morstan
4 inspector Baynes
9 Mycroft Holmes
01 01 02 00 00 01 01 02
02 01 00 01 00 01 01 01
01 01 02 02 01 00 01 00
00 01 01 02 02 01 00 00
█
```

Lorsqu'un joueur clique sur "se connecter" pour entrer dans le jeu, l'interface client:








```
wang@ubuntu:~/s7/os/sh13_etu$ ./sh13 127.0.0.1 3200 127.0.0.1 5000 AAAA
Sans=0x5618a287b9f0
Creation du thread serveur tcp !
```






























l'interface du serveur:

```
00 01 01 02 02 01 00 00
Received packet from 127.0.0.1:58358
Data: [C 127.0.0.1 5000 AAAA
]
COM=C ipAddress=127.0.0.1 port=5000 name=AAAA
0: 127.0.0.1 05000 AAAA
id=0
█
```

et l'interface du jeu:

SDL2 SH13

	 5	 5	 5	 5	 4	 3	 3	 3
.AAAA								
-								
-								
-								

 	Sebastian Moran	
  	Irene Adler	
  	Inspector Lestrade	
  	Inspector Gregson	
 	Inspector Baynes	
 	Inspector Bradstreet	
  	Inspector Hopkins	
  	Sherlock Holmes	
  	John Watson	
  	Mycroft Holmes	
 	Mrs. Hudson	
 	Mary Morstan	
 	James Moriarty	

6.2 Progression du jeu

Lorsque les quatre joueurs sont connectés, le serveur distribue les cartes aux joueurs.

```

Received packet from 127.0.0.1:58358
Data: [C 127.0.0.1 5000 AAAA
]

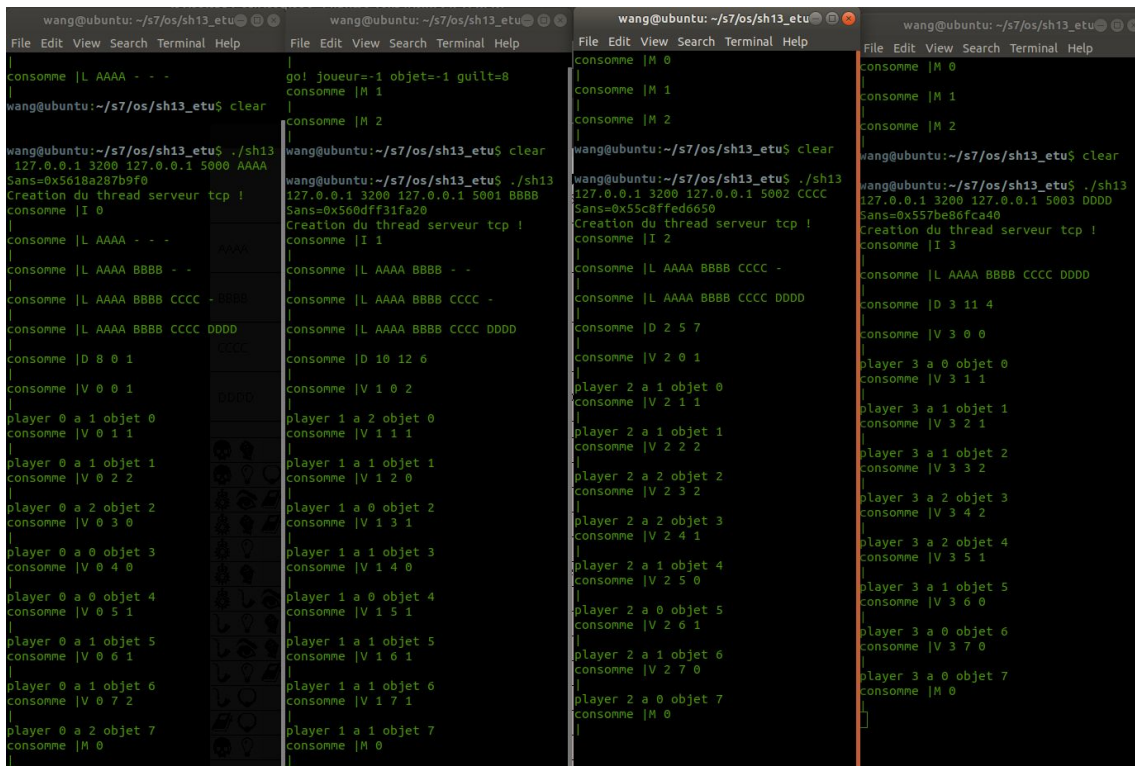
COM=C ipAddress=127.0.0.1 port=5000 name=AAAA
0: 127.0.0.1 05000 AAAA
id=0
Received packet from 127.0.0.1:58366
Data: [C 127.0.0.1 5001 BBBB
]

COM=C ipAddress=127.0.0.1 port=5001 name=BBBB
0: 127.0.0.1 05000 AAAA
1: 127.0.0.1 05001 BBBB
id=1
Received packet from 127.0.0.1:58374
Data: [C 127.0.0.1 5002 CCCC
]

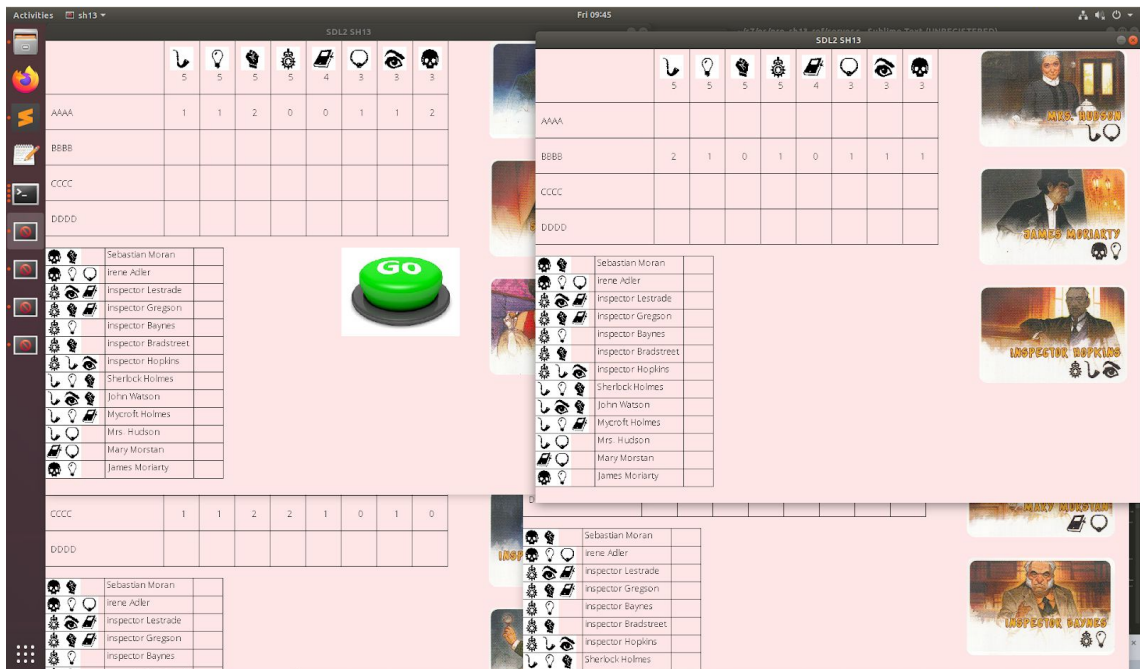
COM=C ipAddress=127.0.0.1 port=5002 name=CCCC
0: 127.0.0.1 05000 AAAA
1: 127.0.0.1 05001 BBBB
2: 127.0.0.1 05002 CCCC
id=2
Received packet from 127.0.0.1:58384
Data: [C 127.0.0.1 5003 DDDD
]

COM=C ipAddress=127.0.0.1 port=5003 name=DDDD
0: 127.0.0.1 05000 AAAA
1: 127.0.0.1 05001 BBBB
2: 127.0.0.1 05002 CCCC
3: 127.0.0.1 05003 DDDD
id=3

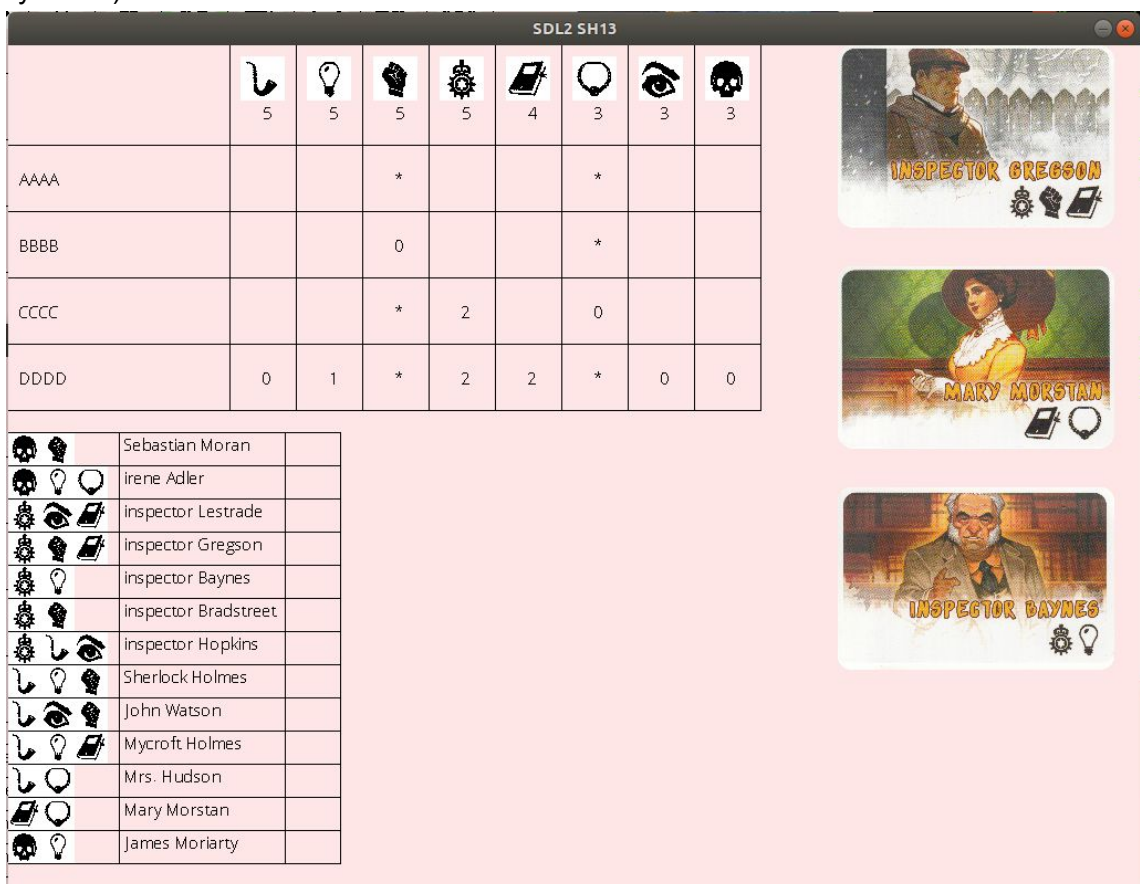
```



Après cela, l'interface de jeu des joueurs devient comme le montre la figure ci-dessous, et le joueur avec le bouton "go" commence sa partie.





Enquête 1 (Choisissez un symbole et demandez aux autres joueurs qui ont ce symbole) fonctionne bien.






Enquête 2 (choisissez un autre joueur et demandez-lui combien de symboles spécifiques il possède) fonctionne bien.

SDL2 SH13

	 5	 5	 5	 5	 4	 3	 3	 3
AAAA	1	1	2	0	0	1	1	2
BBBB								
CCCC				2				
DDDD								





	Sebastian Moran	
	Irene Adler	
	Inspector Lestrade	
	Inspector Gregson	
	Inspector Baynes	
	Inspector Bradstreet	
	Inspector Hopkins	
	Sherlock Holmes	
	John Watson	
	Mycroft Holmes	
	Mrs. Hudson	
	Mary Morstan	
	James Moriarty	

6.3 Accusation

Lorsqu'un joueur fait une fausse accusation, le serveur affichera ce qui suit:

```
joueurCourant est: 3
consomme |
En raison de mauvaise accusation
Le joueur DDDD a été éliminé
```

Après cela, le serveur passe son tour de jeu.

Lorsqu'un joueur fait la bonne accusation ou que trois joueurs ont fait la mauvaise accusation, le jeu se termine et le serveur affiche ce qui suit:

```
joueurCourant est: 0
consomme |
The gagnant selected is correct: Mycroft Holmes
The winner is:AAAA
```



```
wang@ubuntu: ~/s7/os/sh13_etu
File Edit View Search Terminal Help
player 3 a 100 objet 4
consomme |M 3
|
joueurCourant est: 3
consomme |
En raison de mauvaise accusation
Le joueur DDDD a été éliminé
|
consomme |M 0
|
joueurCourant est: 0
go! joueur=-1 objet=-1 guilt=9
consomme |
The gagnant selected is correct: Mycroft Holmes
The winner is:AAAA
|
consomme |M -1
|
joueurCourant est: -1
wang@ubuntu:~/s7/os/sh13_etu$

wang@ubuntu:~/s7/os/sh13_etu
File Edit View Search Terminal Help
player 3 a 100 objet 4
consomme |M 3
|
joueurCourant est: 3
go! joueur=-1 objet=-1 guilt=-1
consomme |
En raison de mauvaise accusation
Le joueur DDDD a été éliminé
|
consomme |M 0
|
joueurCourant est: 0
consomme |
The gagnant selected is correct: Mycroft Holmes
The winner is:AAAA
|
consomme |M -1
|
joueurCourant est: -1
wang@ubuntu:~/s7/os/sh13_etu$

wang@ubuntu:~/s7/os/sh13_etu
File Edit View Search Terminal Help
0: 127.0.0.1 05000 AAAA
ld=0
Received packet from 127.0.0.1:59726
Data: [C 127.0.0.1 5001 BBBB
]
COM=C ipAddress=127.0.0.1 port=5001 name=BBBB
0: 127.0.0.1 05000 AAAA
1: 127.0.0.1 05001 BBBB
ld=1
Received packet from 127.0.0.1:59734
Data: [C 127.0.0.1 5002 CCCC
]
COM=C ipAddress=127.0.0.1 port=5002 name=CCCC
0: 127.0.0.1 05000 AAAA
1: 127.0.0.1 05001 BBBB
2: 127.0.0.1 05002 CCCC
ld=2
Received packet from 127.0.0.1:59744
Data: [C 127.0.0.1 5003 DDDD
]
COM=C ipAddress=127.0.0.1 port=5003 name=DDDD
1: 127.0.0.1 05001 BBBB
2: 127.0.0.1 05002 CCCC
3: 127.0.0.1 05003 DDDD
ld=3
Received packet from 127.0.0.1:59838
Data: [S 0 2 2
]
Received packet from 127.0.0.1:59864
Data: [O 2 4
]
Received packet from 127.0.0.1:59906
Data: [G 3 0
]
Received packet from 127.0.0.1:59924
Data: [G 0 9
]
^C
wang@ubuntu:~/s7/os/sh13_etu$
```