**PROJET PROGRAMMATION SYSTÈME SH13**

**Description générale de projet**

L’objectif de ce projet est de fabriquer une application pour le jeu Sherlock 13.

Il existe 13 cartes, 4 joueurs, 3 cartes sont distribuées par joueur et la 13e carte est celle du coupable à trouver. Le joueur qui trouve la carte gagnera.

Tout d'abord, chaque carte possède des symboles de différents types : pipe, ampoule, main, couronne, livre, crâne, œil ou bien collier.

Chaque joueur sait combien de symboles il y a pour chaque catégorie ainsi que combien il possède de symboles.

A chaque tour, le joueur peut faire une action parmi 3 :

* Accuser une personne, dans ce cas :
* S'il a tort (un des joueur possède la carte), il perd mais la partie continue.
* S'il a raison, il gagne la partie.
* Demander à tous qui a tel symbole ?
* Tous les joueurs doivent répondre par oui ou non (pas le nombre de symbole).
* Demander à un des joueurs combien il a tel symbole.
* Le joueur doit répondre.

On peut noter les symboles de chacun des joueurs sur le tableau ci-dessous. Grâce au tableau, on peut trouver le coupable :

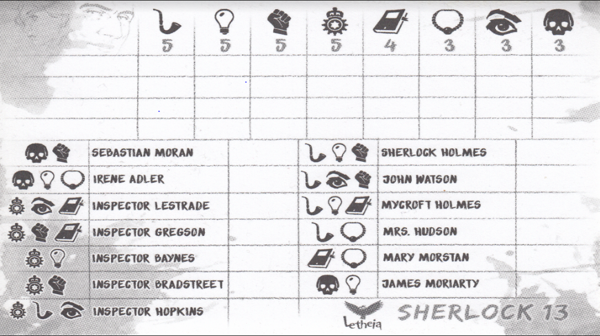


Figure 1 : Tableau des différentes symboles de chaque joueur

Dans ce projet, on utilise une architecture client/serveur, chaque client doit se connecter au serveur pour pouvoir jouer. Ensuite, le serveur va répartir les communications (message à un utilisateur ou à tout le monde). Il y a une petite particularité, c’est que comme le client doit aussi envoyer des informations, il sera également serveur. Le client (un peu serveur également) sera codé par le fichier **sh13.c** et le serveur par **server.c**.

**Explication du code server.c**

Ce serveur sera la base de données contenant les infos de toutes les cartes du jeu, et de ce que possède le joueur. Tous les clients communiquent avec lui, mais pas entre eux directement.

On lance le serveur avec cette commande :

./server <port\_d’ecoute\_du\_serveur>

D’autres seront expliquées plus loin.

Le code de ce serveur comporte plusieurs fonctions, des fonctions pour organiser les cartes et données du jeu.

Une fonction pour envoyer un message au client. Et une fonction pour envoyer un message dit “broadcast” c’est-à-dire à tous les clients connectés.

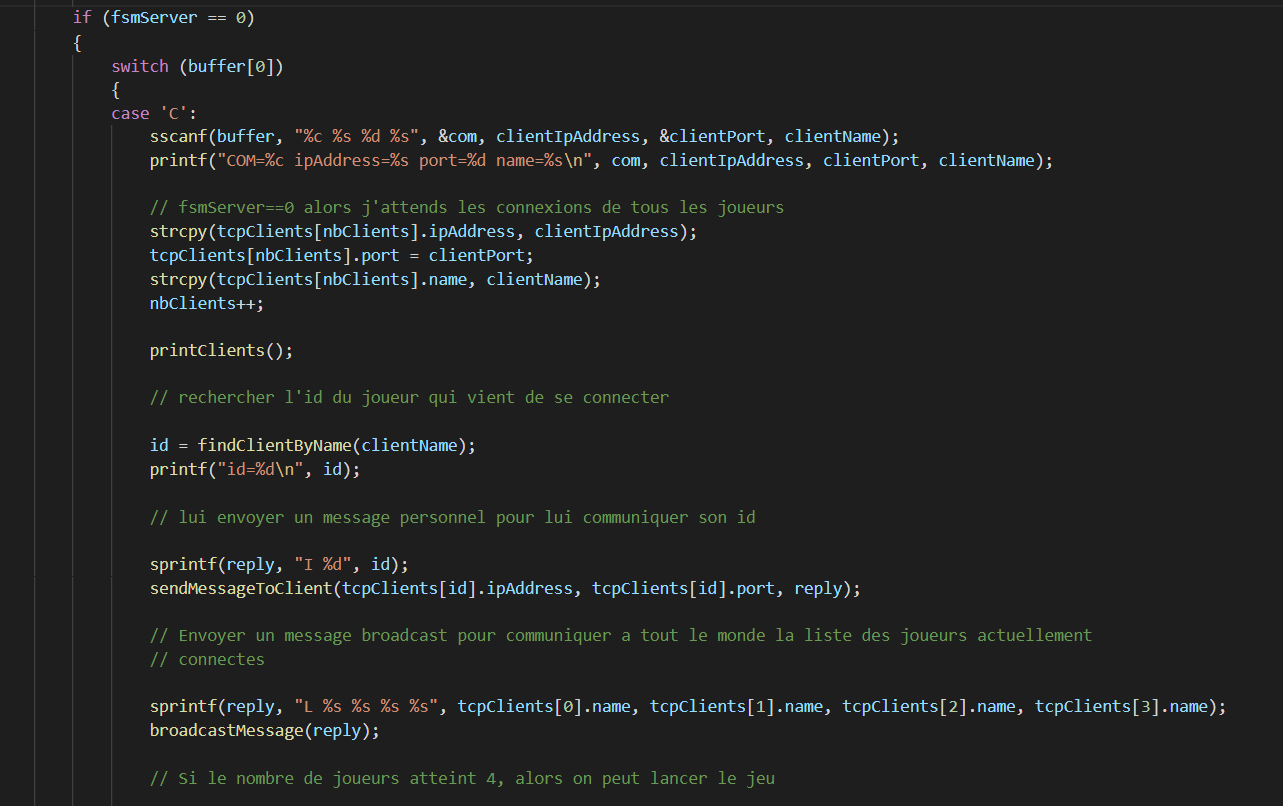
Parmis les variables globales importantes dans ce code on trouve :

* *tableCartes[4][8]* qui contient le nombre de symboles obtenus (8 symboles différents) par chaque joueur (4 joueurs).

Le début de la fonction *main()* est le paramétrage de la connexion TCP, ensuite le programme prépare le jeu (cartes, symboles, etc…).

Ensuite une boucle infinie va recevoir des messages de clients, et y répondre en fonction de l’état du jeu.

Tout d’abord le jeu va attendre la connexion de tous les joueurs. Cela va être caractérisé par un drapeau *fsmserver* auquel la valeurs 0 a été affectée.



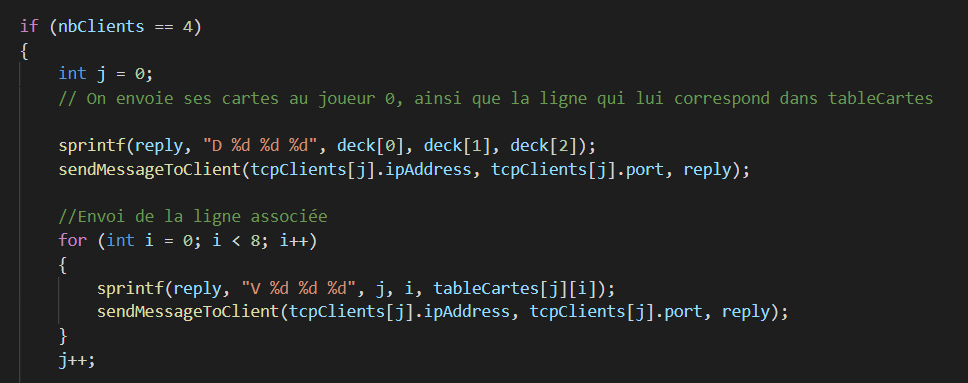
Le message étant stocké dans une chaîne *buffer*, on va tester la valeur du premier caractère nous donnant le type de contenu du message.

**Cas buffer[0]==’C’ :**

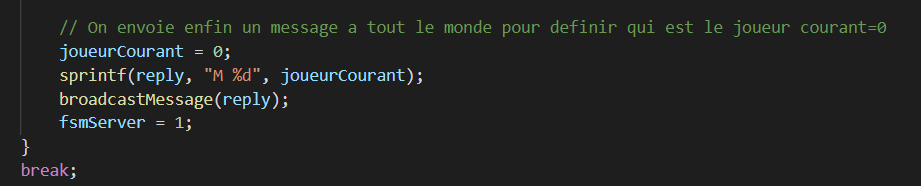
Le message est alors un message de connexion d’un nouveau client. On va alors enregistrer les informations du client/joueur dans notre structure *tcpClient* : nom, adresse IP et port.

On répond alors au client en lui envoyant son ID et on envoie un message broadcast à tous les clients connectés pour leur notifier l'arrivée d’un nouveau joueur.

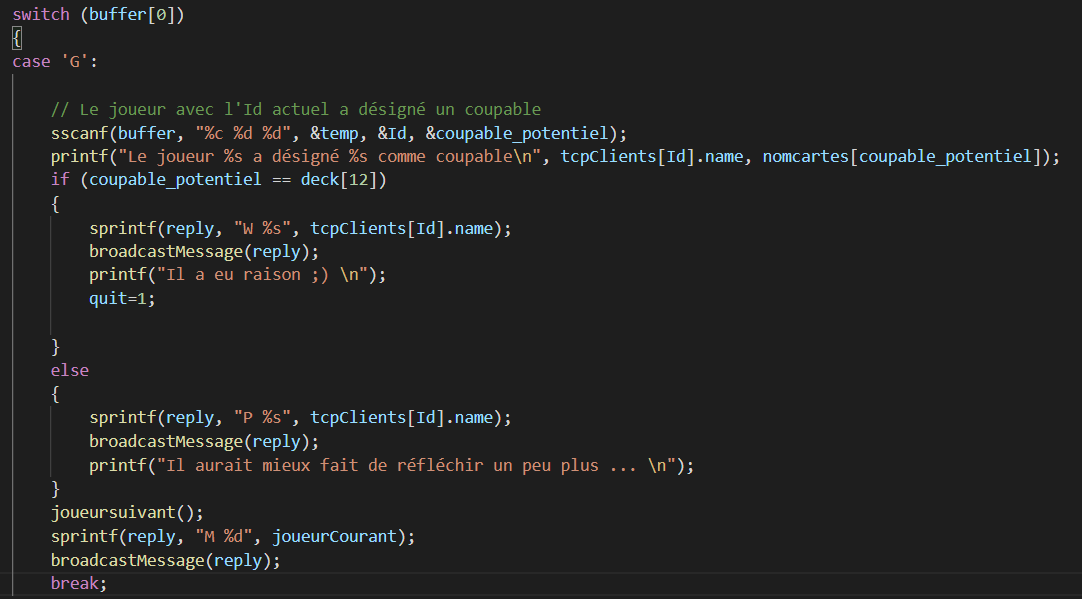
Lorsque 4 client sont connectés, on peut démarrer la partie.



On envoie ses cartes à chaque joueur ainsi que la ligne qui lui est associée (contenant les symboles qu’obtient le joueur avec ses cartes).



Et on finit par envoyer à tout le monde l’ID du joueur qui a la main, et on passe *fsmServer* à 1 pour dire que les 4 joueurs sont connectés.



Les 4 joueurs sont connectés, les messages arrivant alors concernent les actions des joueurs.  
3 actions sont possibles :

**Cas buffer[0]==’G’**

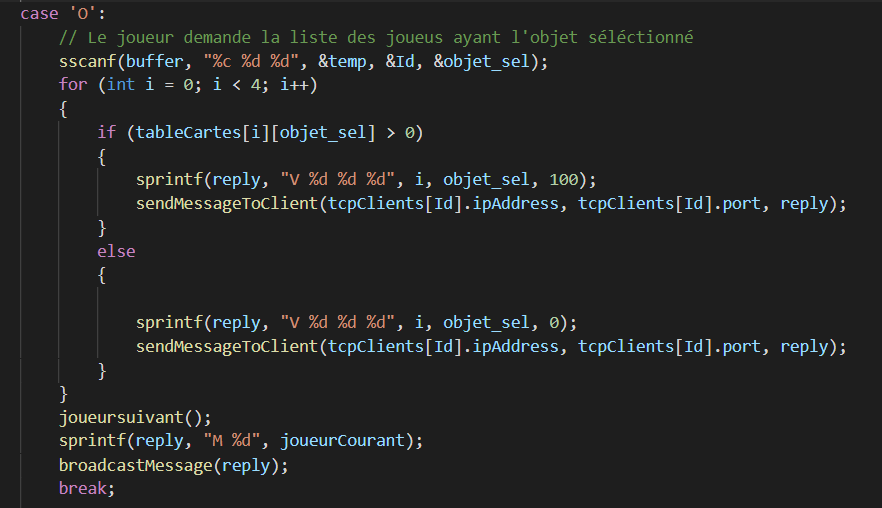
Ce message correspond à la désignation d’un coupable par un joueur.

Il est sous la forme : “C ID\_du\_joueur Valeur\_carte\_perso\_coupable”

Si le coupable potentiel est le même que la dernière carte du paquet (non distribués donc par définition la carte du coupable) alors le joueur a gagné, on envoie alors un message broadcast donnant le nom du vainqueur et on quitte le programme.

Sinon on envoie de même un message broadcast mais disant le nom du perdant et le jeu continue.

**Cas buffer[0]==’O’**

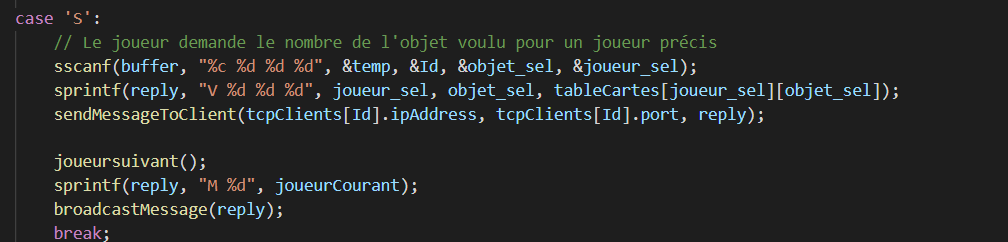


Dans ce cas, le joueur demande qui possède un tel symbole (variable *objet\_sel*).

Le message est sous la forme : “O ID\_du\_joueur symbole\_souhaité”

On fait une boucle pour balayer les lignes du tableau à 2 dimensions contenant la liste des symboles pour chaque joueur, si un joueur possède au moins un symbole, alors le serveur envoi la valeur 100, sinon 0.

**Cas buffer[0]==’S’**



Ici on demande au serveur le nombre d’un symbole spécifique appartenant à un joueur donné.

Le message est sous la forme : “C ID\_du\_joueur symbole\_souhaité joueur\_selectioné”

**Explication du code sh13.c**

Le code est séparé en 3 parties, une partie réseau avec un thread pour créer un serveur de communication avec le serveur du jeu et une fonction pour envoyer un message. Une partie qui va gérer les messages à envoyer selon les actions du joueur sur l’interface graphique et une dernière qui va s’occuper de créer l’interface graphique. L’objectif de ce projet étant d’étudier les appels systèmes, l’utilisation de thread et la programmation réseaux, la partie sur la création de l’interface graphique ne sera pas explicitée.

On lance donc l’application sh13 avec cette commande :

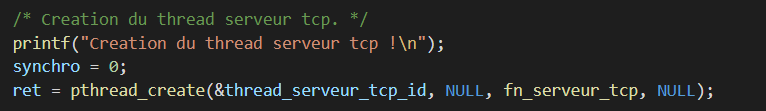
./sh13 <Adresse\_du\_serveur> <port\_d'écoute\_du\_serveur> <Adresse\_du\_client> <port\_ou\_ecouter\_les\_reponse\_du\_client>

Cela va permettre de récupérer dans la variable *argv* les infos nécessaires à la communication réseau.

Partie réseau :

Elle se compose de deux fonctions et de quelques lignes dans la fonction *main()*

Les lignes dans *main()* vont créer un thread qui va appeler la fonction *fn\_serveur\_tcp*.



Voici la fonction :

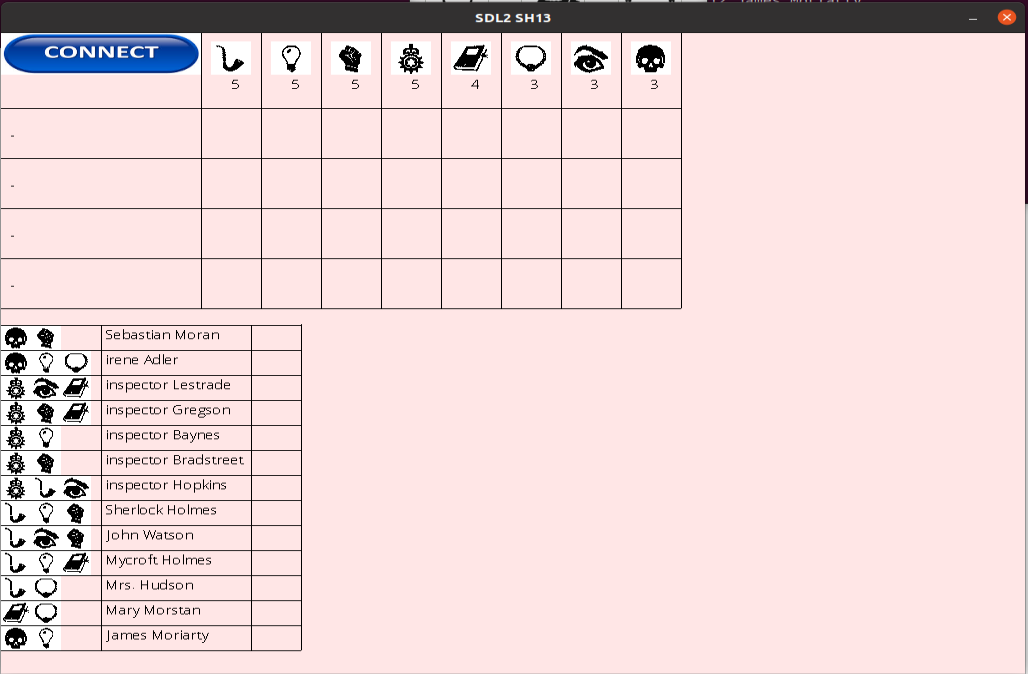


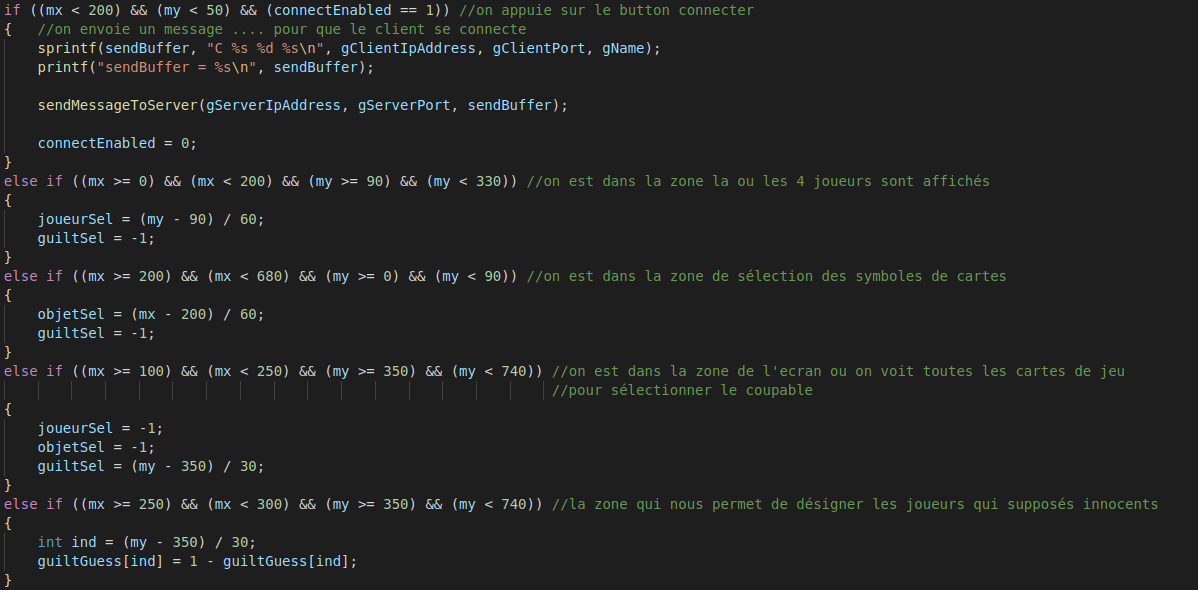
Cette fonction va établir une communication TCP avec le serveur du jeu, puis entrer dans une boucle infinie qui, à la fin de chaque boucle, va passer la variable synchro à 1 puis attendre qu'elle passe à 0 (pour être sûr que le programme va traiter le message).

Ensuite on a une fonction sendmessageToServer qui va demander une connexion vers l’hote et lui envoyer un message celle-ci est similaire à celle du serveur.

Partie gestion des message :

Lors du lancement de l’application client cette interface s’affiche :





En cliquant sur le bouton “Connect”, on envoie un message au serveur en désignant le joueur qui va se connecter, et ce dernier est identifié par son adresse IP, Port et Nom donnée comme argument de lancement, dans le message transmis en ajoutant les différentes valeurs dans la chaîne de caractère *senBuffer* :

Pour ensuite envoyer cette chaîne au serveur.

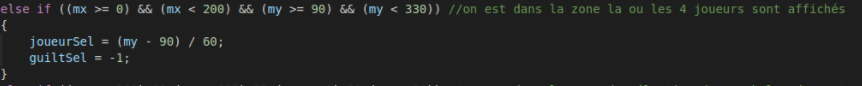
Ensuite l’interface graphique du client va changer pour afficher un tableau avec le nom des joueurs connectés. Une fois que 4 joueurs sont connectés, le serveur va envoyer alors les informations à chaque joueur c'est-à-dire ses cartes, et sa ligne de symboles



Après selon la zone de l’écran sélectionné par la souris on peut en cliquant effectuer plusieurs actions

- **Zone d’affichage des 4 joueurs :**

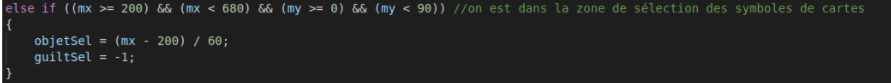
****

****

En cliquant sur le nom d’un joueur, on va affecter un numéro de joueur à la variable *joueursel.* Ce numéro est calculé avec la position de la souris.

- **Zone de la sélection des symboles de cartes :**

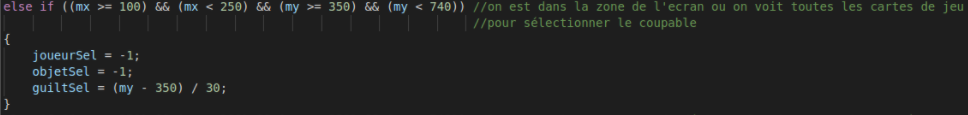
****



En sélectionnant un symbole on va affecter de la même manière que précédemment une valeur correspondant à un symbole.

**- Zone de sélection de coupable :**

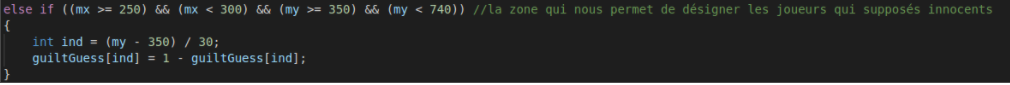
****



Ici on peut sélectionner un nom parmi les 13 personnages du jeu pour désigner le coupable.

**- Zone qui nous permet d’identifier les joueurs supposés innocents :**

****

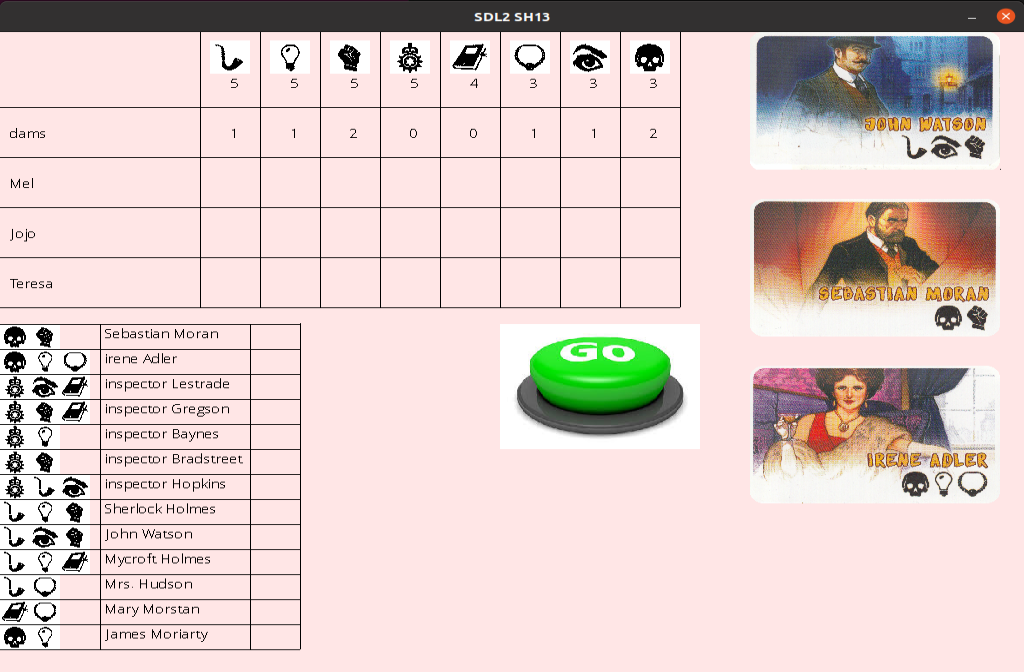


Cette zone permet de sélectionner et d’afficher une croix devant les noms de personnages écartés en tant que coupable.

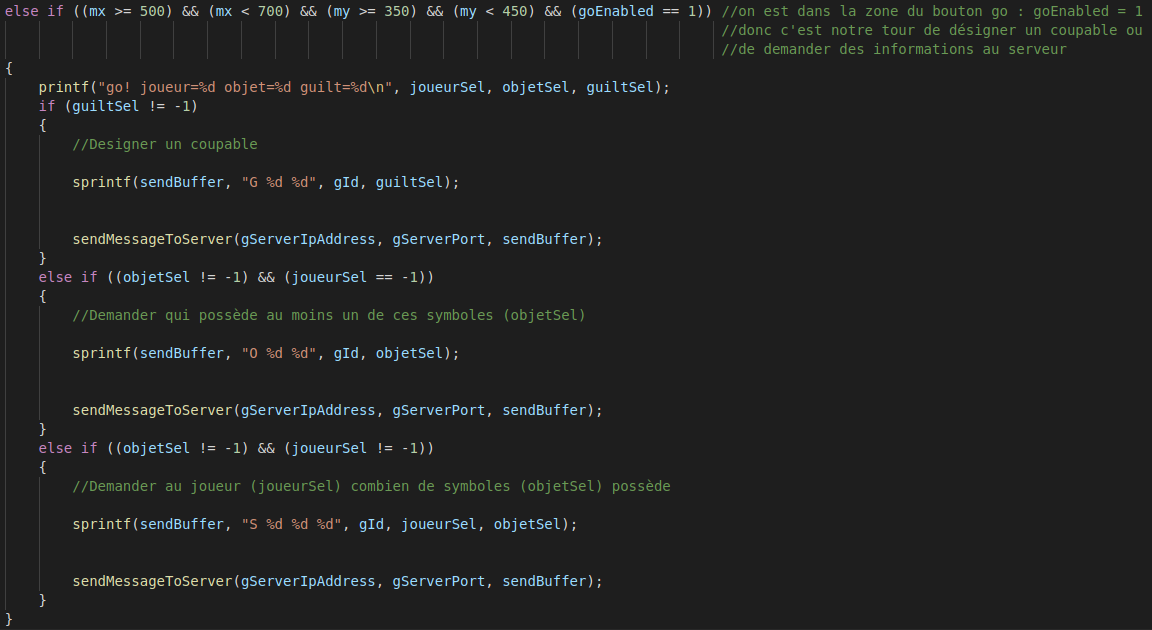
Comme il sera expliqué plus tard, lorsque c’est notre tour de jouer la variable goEnabled sera à 1 et un bouton GO sera affiché sur notre interface. A l’appui de “GO” on enverra notre choix d’action au serveur, et ce sera au tour du joueur suivant.

En appuyant sur le bouton go, il sera donc notre tour de désigner le “guilt” ou bien de demander des informations au serveur.

Si la souris est sur la zone du bouton GO et qu’il y a un clique, alors selon les variables : *joueurSel, objetSel et guiltSel l’action sera différentes.*

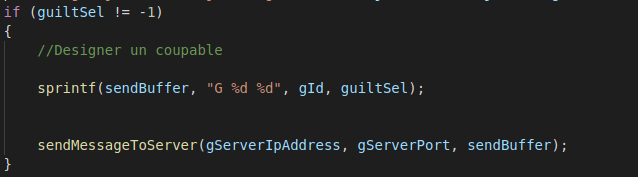


Ces variables sont par défaut à -1 (aucun objet, joueur ou coupable n’est sélectionné).

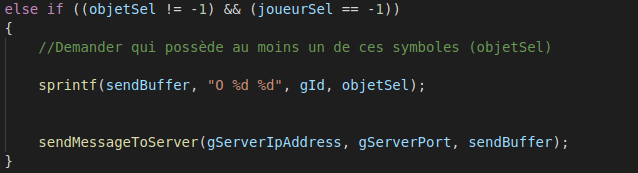


On est capable désormais choisir parmi ces trois action :

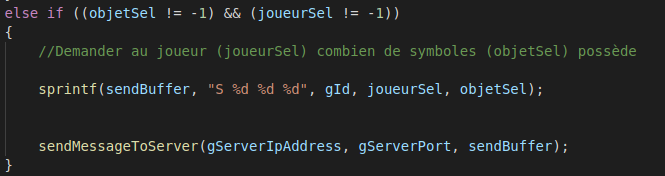
* Si *guiltSel* est différent de -1, on peut désigner un coupable avec le contenu de la variable guiltSel qui désigne le numéro du personnage choisi. On transmet ensuite cela au serveur avec le caractère G, l’Id du client et *guiltSel*:



* Si *objetSel* est différent de -1 mais *joueurSel* vaut -1, alors on demande à tous si ils ont le symbole désigné par *objetSel*. On envoi un message avec le caractère O, l’Id du client et *objetSel* :



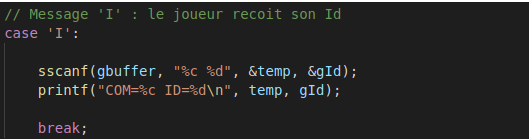
* Enfin *joueurSel* et *objetSel* sont différent de -1, on demande alors au joueur *joueurSel* combien il a de symboles *ObjetSel* :



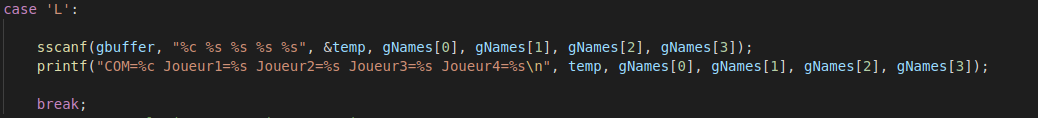
Autrement, on passe le tour à un autre joueur.

A la réception d’un nouveau message (synchro == 1), et selon la première lettre (gbuffer[0]) qui désigne le type de message reçu, on peut citer plusieurs cas :

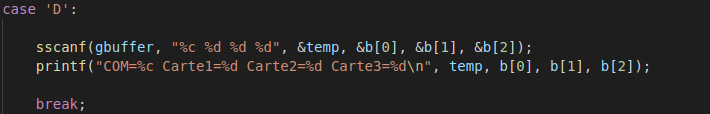
* **Cas gbuffer[0] = ‘I’ :** le joueur reçoit son Id (gId) :



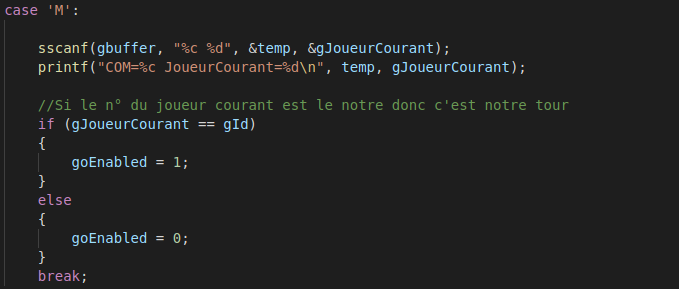
* **Cas gbuffer[0] = ‘L’ :** le joueur reçoit la liste des joueurs (gName[0], gName[1], gName[2], gName[3]) :



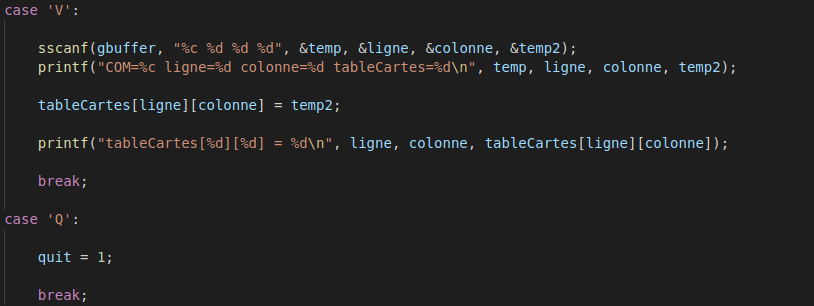
* **Cas gbuffer[0] = ‘D’ :** Le joueur reçoit ces trois cartes (b[0], b[1], b[2]) :



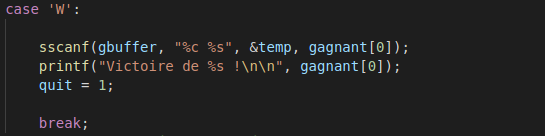
* **Cas gbuffer[0] = ‘M’ :** le joueur reçoit le numéro du joueur courant (gJoueurCourant). Au cas ou ce numéro est le nôtre, goEnabled = 1 :



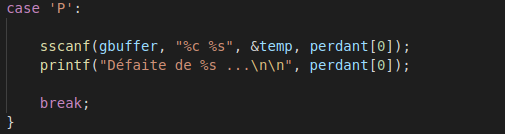
* **Cas gbuffer[0] = ‘V’ :** le joueur reçoit une valeur de tableCartes représentée par le numéro de ligne et le numéro de colonne (ligne, colonne) :



* **Cas gbuffer[0] = ‘W’ :** déclarer la victoire d’un joueur (gagnant[0]) :



* **Cas gbuffer[0] = ‘P’ :** déclarer la défaite d’un joueur (perdant[0]) :



Case I,L,D,M on utilise les fonctions sscanf qui permet de lire et d’extraire une chaîne de caractères, on peut ainsi connaître l’Id, les noms etc.. des joueurs puis les afficher avec un printf.

Pour le case M on rajoute une condition if permettant d’afficher le bouton go , le bouton go signifie que l’on est le joueur courant .

En ce qui concerne la case V, Chaque joueur possède une table « tableCarte » qui désigne les objets en fonction des joueurs. De plus, chaque joueur est représenté sur une ligne et une colonne.