

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 :

			5		5		5		5		4		3		3		3
		SEBASTIAN MORAN									SHERLOCK HOLMES						
		IRENE ADLER									JOHN WATSON						
		INSPECTOR LESTRADE									MYCROFT HOLMES						
		INSPECTOR GREGSON									MRS. HUDSON						
		INSPECTOR BAYNES									MARY MORSTAN						
		INSPECTOR BRADSTREET									JAMES MORIARTY						
		INSPECTOR HOPKINS															

Lethicia

SHERLOCK 13

Lethicia

SHERLOCK 13

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.

Parmi 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 valeur 0 a été affectée.

```

if (fsmServer == 0)
{
    switch (buffer[0])
    {
        case 'C':
            sscanf(buffer, "%c %s %d %s", &com, clientIpAddress, &clientPort, clientName);
            printf("COM=%c ipAddress=%s port=%d name=%s\n", com, clientIpAddress, clientPort, clientName);

            // fsmServer==0 alors j'attends les connexions de tous les joueurs
            strcpy(tcpClients[nbClients].ipAddress, clientIpAddress);
            tcpClients[nbClients].port = clientPort;
            strcpy(tcpClients[nbClients].name, clientName);
            nbClients++;

            printClients();

            // rechercher l'id du joueur qui vient de se connecter

            id = findClientByName(clientName);
            printf("id=%d\n", id);

            // lui envoyer un message personnel pour lui communiquer son id

            sprintf(reply, "I %d", id);
            sendMessageToClient(tcpClients[id].ipAddress, tcpClients[id].port, reply);

            // Envoyer un message broadcast pour communiquer a tout le monde la liste des joueurs actuellement
            // connectes

            sprintf(reply, "L %s %s %s %s", tcpClients[0].name, tcpClients[1].name, tcpClients[2].name, tcpClients[3].name);
            broadcastMessage(reply);
    }
}

```

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.

```

if (nbClients == 4)
{
    int j = 0;
    // On envoie ses cartes au joueur 0, ainsi que la ligne qui lui correspond dans tableCartes

    sprintf(reply, "D %d %d %d", deck[0], deck[1], deck[2]);
    sendMessageToClient(tcpClients[j].ipAddress, tcpClients[j].port, reply);

    //Envoi de la ligne associée
    for (int i = 0; i < 8; i++)
    {
        sprintf(reply, "V %d %d %d", j, i, tableCartes[j][i]);
        sendMessageToClient(tcpClients[j].ipAddress, tcpClients[j].port, reply);
    }
    j++;
}

```

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).

```

    // On envoie enfin un message a tout le monde pour definir qui est le joueur courant=0
    joueurCourant = 0;
    sprintf(reply, "M %d", joueurCourant);
    broadcastMessage(reply);
    fsmServer = 1;
}
break;

```

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.

```

switch (buffer[0])
{
case 'G':
    // Le joueur avec l'Id actuel a désigné un coupable
    sscanf(buffer, "%c %d %d", &temp, &Id, &coupable_potentiel);
    printf("Le joueur %s a désigné %s comme coupable\n", tcpClients[Id].name, nomcartes[coupable_potentiel]);
    if (coupable_potentiel == deck[12])
    {
        sprintf(reply, "W %s", tcpClients[Id].name);
        broadcastMessage(reply);
        printf("Il a eu raison ;) \n");
        quit=1;
    }
    else
    {
        sprintf(reply, "P %s", tcpClients[Id].name);
        broadcastMessage(reply);
        printf("Il aurait mieux fait de réfléchir un peu plus ... \n");
    }
    joueursuivant();
    sprintf(reply, "M %d", joueurCourant);
    broadcastMessage(reply);
    break;
}

```

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'

```

case 'O':
    // Le joueur demande la liste des joueurs ayant l'objet sélectionné
    sscanf(buffer, "%c %d %d", &temp, &Id, &objet_sel);
    for (int i = 0; i < 4; i++)
    {
        if (tableCartes[i][objet_sel] > 0)
        {
            sprintf(reply, "v %d %d %d", i, objet_sel, 100);
            sendMessageToClient(tcpClients[Id].ipAddress, tcpClients[Id].port, reply);
        }
        else
        {
            sprintf(reply, "v %d %d %d", i, objet_sel, 0);
            sendMessageToClient(tcpClients[Id].ipAddress, tcpClients[Id].port, reply);
        }
    }
    joueursuivant();
    sprintf(reply, "M %d", joueurCourant);
    broadcastMessage(reply);
    break;

```

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 envoie la valeur 100, sinon 0.

Cas buffer[0]=='S'

```

case 'S':
    // Le joueur demande le nombre de l'objet voulu pour un joueur précis
    sscanf(buffer, "%c %d %d %d", &temp, &Id, &objet_sel, &joueur_sel);
    sprintf(reply, "v %d %d %d", joueur_sel, objet_sel, tableCartes[joueur_sel][objet_sel]);
    sendMessageToClient(tcpClients[Id].ipAddress, tcpClients[Id].port, reply);

    joueursuivant();
    sprintf(reply, "M %d", joueurCourant);
    broadcastMessage(reply);
    break;

```

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_selectionné"

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*.

```
/* Creation du thread serveur tcp. */  
printf("Creation du thread serveur tcp !\n");  
synchro = 0;  
ret = pthread_create(&thread_serveur_tcp_id, NULL, fn_serveur_tcp, NULL);
```

Voici la fonction :

```
void *fn_serveur_tcp(void *arg)
{
    int sockfd, newsockfd, portno;
    socklen_t clilen;
    struct sockaddr_in serv_addr, cli_addr;
    int n;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
    {
        printf("sockfd error\n");
        exit(1);
    }
    bzero((char *)&serv_addr, sizeof(serv_addr));
    portno = gClientPort;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        printf("bind error\n");
        exit(1);
    }
    listen(sockfd, 5);
    clilen = sizeof(cli_addr);
    while (1)
    {
        newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);
        if (newsockfd < 0)
        {
            printf("accept error\n");
            exit(1);
        }
        bzero(gbuffer, 256);
        n = read(newsockfd, gbuffer, 255);
        if (n < 0)
        {
            printf("read error\n");
            exit(1);
        }
        pthread_mutex_lock(&mutex);
        synchro = 1;
        pthread_mutex_unlock(&mutex);
        while (synchro);
    }
}
```

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'hôte 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` :

```
sprintf(sendBuffer, "C %s %d %s\n", gClientIpAddress, gClientPort, gName);
```

Pour ensuite envoyer cette chaîne au serveur.

```
sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
```


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

SDL2 SH13









	5	5	5	5	4	3	3	3
dams	1	1	2	0	0	1	1	2
Mel		1						
Jojo								
Teresa								


































	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	




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

- Zone d'affichage des 4 joueurs :

SDL2 SH13

	 5	 5	 5	 5	 4	 3	 3	 3
dams	1	1	2	0	0	1	1	2
Mel		1						
Jojo								
Teresa								

 	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	

```









else if ((mx >= 0) && (mx < 200) && (my >= 90) && (my < 330)) //on est dans la zone la ou les 4 joueurs sont affichés
{
    joueurSel = (my - 90) / 60;
    guiltSel = -1;
}

































```




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 :

SDL2 SH13

	 5	 5	 5	 5	 4	 3	 3	 3
dams	1	1	2	0	0	1	1	2
Mel		1						
Jojo								
Teresa								

 	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	

```









else if ((mx >= 200) && (mx < 680) && (my >= 0) && (my < 90)) //on est dans la zone de sélection des symboles de cartes
{
    objetSel = (mx - 200) / 60;
    guiltSel = -1;
}














```


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


- Zone de sélection de coupable :


SDL2 SH13

	 5	 5	 5	 5	 4	 3	 3	 3
dams	1	1	2	0	0	1	1	2
Mel		1						
Jojo								
Teresa								

	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







```









else if ((mx >= 100) && (mx < 250) && (my >= 350) && (my < 740)) //on est dans la zone de l'ecran ou on voit toutes les cartes de jeu
{
    //pour sélectionner le coupable
    joueurSel = -1;
    objetSel = -1;
    guiltSel = (my - 350) / 30;
}



























```




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 :

SDL2 SH13

	 5	 5	 5	 5	 4	 3	 3	 3
dams	1	1	2	0	0	1	1	2
Mel		1						
Jojo								
Teresa								

 	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	

```

else if ((mx >= 250) && (mx < 300) && (my >= 350) && (my < 740)) //la zone qui nous permet de désigner les joueurs qui supposés innocents
{
    int ind = (my - 350) / 30;
    guiltGuess[ind] = 1 - guiltGuess[ind];
}

```




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.


SDL2 SH13

	 5	 5	 5	 5	 4	 3	 3	 3
dams	1	1	2	0	0	1	1	2
Mel								
Jojo								
Teresa								

 	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	







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

```

else if ((mx >= 500) && (mx < 700) && (my >= 350) && (my < 450) && (goEnabled == 1)) //on est dans la zone du bouton go : goEnabled = 1
{
    //donc c'est notre tour de désigner un coupable ou
    //de demander des informations au serveur

    printf("go! joueur=%d objet=%d guilt=%d\n", joueurSel, objetSel, guiltSel);
    if (guiltSel != -1)
    {
        //Designer un coupable

        sprintf(sendBuffer, "G %d %d", gId, guiltSel);

        sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
    }
    else if ((objetSel != -1) && (joueurSel == -1))
    {
        //Demander qui possède au moins un de ces symboles (objetSel)

        sprintf(sendBuffer, "O %d %d", gId, objetSel);

        sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
    }
    else if ((objetSel != -1) && (joueurSel != -1))
    {
        //Demander au joueur (joueurSel) combien de symboles (objetSel) possède

        sprintf(sendBuffer, "S %d %d %d", gId, joueurSel, objetSel);

        sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
    }
}

```


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*:

```
if (guiltSel != -1)
{
    //Designer un coupable

    sprintf(sendBuffer, "G %d %d", gId, guiltSel);

    sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
}
```

- 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* :

```
elseif ((objetSel != -1) && (joueurSel == -1))
{
    //Demander qui possède au moins un de ces symboles (objetSel)

    sprintf(sendBuffer, "O %d %d", gId, objetSel);

    sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
}
```

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

```
else if ((objetSel != -1) && (joueurSel != -1))
{
    //Demander au joueur (joueurSel) combien de symboles (objetSel) possède

    sprintf(sendBuffer, "S %d %d %d", gId, joueurSel, objetSel);

    sendMessageToServer(gServerIpAddress, gServerPort, sendBuffer);
}
```

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*) :

```
case 'I':

    sscanf(gbuffer, "%c %d", &temp, &gId);
    printf("COM=%c ID=%d\n", temp, gId);

    break;
```

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

```
case 'L':

    sscanf(gbuffer, "%c %s %s %s %s", &temp, gNames[0], gNames[1], gNames[2], gNames[3]);
    printf("COM=%c Joueur1=%s Joueur2=%s Joueur3=%s Joueur4=%s\n", temp, gNames[0], gNames[1], gNames[2], gNames[3]);

    break;
```

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

```
case 'D':

    sscanf(gbuffer, "%c %d %d %d", &temp, &b[0], &b[1], &b[2]);
    printf("COM=%c Carte1=%d Carte2=%d Carte3=%d\n", temp, b[0], b[1], b[2]);

    break;
```

- Cas gbuffer[0] = 'M' : le joueur reçoit le numéro du joueur courant (gJoueurCourant). Au cas où ce numéro est le nôtre, goEnabled = 1 :

```
case 'M':

    sscanf(gbuffer, "%c %d", &temp, &gJoueurCourant);
    printf("COM=%c JoueurCourant=%d\n", temp, gJoueurCourant);

    //Si le n° du joueur courant est le notre donc c'est notre tour
    if (gJoueurCourant == gId)
    {
        goEnabled = 1;
    }
    else
    {
        goEnabled = 0;
    }
    break;
```

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

```
case 'V':

    sscanf(gbuffer, "%c %d %d %d", &temp, &ligne, &colonne, &temp2);
    printf("COM=%c ligne=%d colonne=%d tableCartes=%d\n", temp, ligne, colonne, temp2);

    tableCartes[ligne][colonne] = temp2;

    printf("tableCartes[%d][%d] = %d\n", ligne, colonne, tableCartes[ligne][colonne]);

    break;

case 'Q':

    quit = 1;

    break;
```

- Cas gbuffer[0] = 'W' : déclarer la victoire d'un joueur (gagnant[0]) :


```

case 'W':

    sscanf(gbuffer, "%c %s", &temp, gagnant[0]);
    printf("Victoire de %s !\n\n", gagnant[0]);
    quit = 1;

    break;

```

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

```

case 'P':

    sscanf(gbuffer, "%c %s", &temp, perdant[0]);
    printf("Défaite de %s ...!\n\n", perdant[0]);

    break;
}

```

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.