22/12/2023

BELASSEL Meryem
NICOLLE Thomas

Rapport du projet S&R

6 qui prend

● Introduction

Le jeu choisi pour ce projet est le 6 qui prend, un jeu de stratégie et de hasard où le but est de placer ses cartes dans les rangées sans poser la sixième, ni la plus petite, pour éviter de ramasser des têtes de bœuf. Pour remporter la partie, il faut posséder le moins de têtes de bœuf possible.
Le projet consiste à développer le jeu en utilisant principalement les langages awk/shell et/ou C. Le jeu se compose de quatre types de processus : le gestionnaire du jeu (également nommé le *maître du jeu*), le joueur humain, le joueur robot et le gestionnaire des statistiques. Le projet propose également une fonctionnalité avancée: le jeu en réseau.
Par la suite, nous allons expliquer les diverses classes ainsi que les algorithmes utilisés. En plus, nous allons décrire comment le jeu se déroule et fonctionne.

● Makefile

Le fichier Makefile contient des lignes qui définissent les *dépendances* entre les *fichiers sources* et les *fichiers objets*, ainsi que les commandes à exécuter pour générer les fichiers exécutables.

Voici une explication de chaque ligne du fichier Makefile :

- **all** : gj jh indique que la cible all dépend des cibles gj et jh, c'est-à-dire qu'il faut compiler ces deux programmes pour obtenir le projet complet. C'est la règle par défaut qui est exécutée quand on tape make sans argument.
- **gestionnaireDeStats.o : gestionnaireDeStats.c** indique que le fichier objet **gestionnaireDeStats.o** dépend du fichier source **gestionnaireDeStats.c**, c'est-à-dire qu'il faut recompiler le fichier source si celui-ci a été modifié. La commande *gcc -c gestionnaireDeStats.c* permet de compiler le fichier source en fichier objet, sans produire de fichier exécutable.
- **Carte.o: Carte.c, jeu.o: jeu.c, joueurRobot.o : joueurRobot.c et joueurHumain.o: joueurHumain.c** sont similaires à la règle précédente, mais pour les fichiers sources Carte.c, jeu.c, joueurRobot.c et joueurHumain.c.
- **gestionnaireDeJeu.o: gestionnaireDeJeu.c** indique que le fichier objet **gestionnaireDeJeu.o** dépend du fichier source **gestionnaireDeJeu.c,** et la commande ***gcc -c gestionnaireDeJeu.c*** permet de le compiler.
- **Gj** : **gestionnaireDeJeu.o joueurRobot.o jeu.o Carte.o gestionnaireDeStats.o** indique que le fichier exécutable gj dépend des fichiers objets **gestionnaireDeJeu.o**, **joueurRobot.o, jeu.o, Carte.o** et **gestionnaireDeStats.o**, c'est-à-dire qu'il faut relier ces fichiers objets pour générer le fichier exécutable. La commande **gcc -o gj**

gestionnaireDeJeu.o joueurRobot.o jeu.o Carte.o gestionnaireDeStats.o permet de réaliser cette opération, en spécifiant le nom du fichier exécutable avec l'option ***-o***.

- **Jh** : **joueurHumain.o Carte.o** indique que le fichier exécutable jh dépend des fichiers objets **joueurHumain.o** et **Carte.o**, et la commande ***gcc -o jh joueurHumain.o Carte.o*** permet de les relier pour produire le fichier exécutable.
- **clean**: indique une cible spéciale qui ne dépend d'aucun fichier, mais qui permet d'effectuer une opération de nettoyage. La commande ***rm -f *.o gj jh*** permet de supprimer tous les fichiers objets et les fichiers exécutables du projet, pour libérer de l'espace disque. Cette ligne est exécutée quand on tape **make clean.**

Classes et algorithmes

A. Gestionnaire de jeu

Le rôle du gestionnaire de jeu est dans un premier temps d'établir le nombre de joueurs dans une partie, ainsi que le nombre de robots; si un seul joueur désire jouer, un robot est automatiquement créé. Ensuite il va attendre la connexion de tous les joueurs afin de débuter la partie; un joueur est associé à un thread. Une fois chaque joueur connecté, ceux-ci attendent le feu vert du maître de jeu (MJ) qui une fois après avoir créé le paquet, le mélange et distribue les cartes puis redonne la main au thread de connexion de joueur qui va envoyer les règles du jeu et redonner la main au maître du jeu. Il va par la suite préparer le plateau du jeu en posant les cartes du paquet sur les 4 rangées. Juste après, le maître du jeu attend que l'ensemble des joueurs choisissent leurs cartes à jouer, puis les placent dans un tableau. Une fois que les choix sont réalisés, le maître du jeu sort ce tableau pour établir l'ordre du jeu, en fonction de la valeur numérique la plus faible. Dans la suite, le maître du jeu libèrent les threads de connexion de joueur en ordre croissant, pour que les joueurs puissent sélectionner la rangée pour poser la carte précédemment choisie. Une fois que tous les joueurs ont joué, le maître du jeu incrémente le nombre de tours.
Le jeu prend fin lorsque les 10 cartes sont épuisées ou bien au moins un joueur atteint un score de 66 têtes de bœuf. Il attendra que tous les threads se terminent pour s'arrêter.

B. Jeu

- **initRangee(Table *t, Carte *ct, int l)** : Cette fonction initialise une rangée du plateau de jeu. Elle place la carte *ct* à la première position de la rangée *l* et remplit le reste de la rangée avec des cartes vides.
- **lastCarte(Table *t, int rangee)** : Cette fonction renvoie la dernière carte non vide de la rangée spécifiée.
- **nbrCarteR(Table *t, int rangee)** : Cette fonction renvoie le nombre de cartes non vides dans la rangée spécifiée.
- **triCroissantP(Proposition t[], int nbrJoueurs)** : Cette fonction sort un tableau de propositions en ordre croissant en fonction du numéro de la carte dans chaque proposition. Elle utilise un algorithme de *tri à bulles*.
- **compterTeteDeBoeuf(Table *t, int ligne, Joueur *j)** : Cette fonction ajoute le nombre total de têtes de bœuf dans une rangée spécifiée au score de pénalité d'un joueur.
- **poserCarteTable(Table *t, int indiceCarte, int rangee, Joueur *j)** : Cette fonction tente de poser une carte d'un joueur sur une rangée spécifiée du plateau. Si la

rangée est complète, elle compte les têtes de bœuf dans la rangée, réinitialise la rangée et place la carte à la première position.

- **distribuerCartes(Paquet *p, Joueur joueurs[], int nbJoueurs, int cartesParJoueur)** : Cette fonction distribue un certain nombre de cartes de la pioche à chaque joueur.
- **afficherMainJoueur(Joueur *j, char *buffer, int taillebuf)** : Cette fonction génère une chaîne de caractères représentant la main d'un joueur et la stocke dans un buffer.
- **initTable(Table *t)** : Cette fonction initialise le plateau de jeu en le remplissant de cartes vides.
- **remplirPlateau(Paquet *p, Table *t)** : Cette fonction remplit le plateau de jeu en plaçant une carte tirée du paquet à la première position de chaque rangée, pour débuter une partie.
- **affichagePlateau(Table *t, char *buffer, int taillebuf)** : Cette fonction génère une chaîne de caractères représentant le *plateau de jeu* et la stocke dans un buffer, nous permettant par la suite d'afficher le plateau d'une façon esthétique.
- **ppCarteRangee(Table *t, int val, Joueur *j)** : Cette fonction vérifie si une carte spécifique dans la main d'un joueur est inférieure à toutes les dernières cartes de chaque rangée sur le plateau, pour que le joueur concerné puisse dans ce cas là, récupérer la rangée qu'il veut et déposer sa carte (appel à la méthode qui suit).
- **poserCarteInfTable(Table *t, int indiceCarte, int rangee, Joueur *j)** : Cette fonction tente de déposer une carte d'un joueur sur une rangée spécifiée du plateau. Si la rangée est complète, elle compte les têtes de bœuf dans la rangée (correspond au fait de récupérer toutes les cartes de la rangée), réinitialise la rangée et place la carte à la première position.
- **verifScore(int classement[max][2], Joueur lesjoueurs[], int nbrJoueurs, char *buf, int taillebuf)** : Cette fonction vérifie les scores de tous les joueurs, les trie et les affiche. Si un joueur a atteint 66 points, la fonction renvoie true, nous permettant par la suite d'arrêter la partie.

A. Joueur Humain

Le joueur humain est le programme client qui va permettre de se connecter au serveur gestionnaire de jeu grâce à son adresse IP et un port spécifié à l'ouverture. C'est la méthode

```
int initJH(char *host, int port);
```

Qui va permettre de faire cela. Ensuite, une fois la communication établie, le client reçoit les actions de la part du serveur et lui envoie ses réponses.

B. Joueur Robot

Le joueur robot participe au jeu comme un joueur humain en tentant de jouer les cartes reçues. Dans sa classe, se trouve les deux méthodes qui permettent de simuler une action :

```
int getNumCarte(Joueur*);
```

Elle permet de récupérer l'indice d'une carte en générant un indice aléatoire compris entre 0 et le nombre de cartes du joueur passé en paramètre. Cela permet de récupérer par la suite la carte située à cet indice dans la main du joueur.

```
int getRangee();
```

Enfin cette méthode permet de retourner le numéro d'une rangée en générant un entier entre 0 et 4 inclus. Cela permet au robot de choisir une rangée lorsqu'il souhaite poser sa carte.

C. Gestionnaire de stats

Pour permettre l'enregistrement des logs d'une partie de jeu, il a fallu écrire un script shell permettant de prendre en paramètre une chaîne de caractère afin d'ajouter cette dernière dans un fichier txt. Si le fichier n'a pas été trouvé, alors ce script va créer le fichier et le remplir. Le gestionnaire de stats est composé d'une méthode

```
void logStats(char* msg);
```

qui va permettre d'exécuter le script avec en paramètre le message "*msg"* à ajouter dans le fichier. Nous appelons cette fonction autant de fois qu'une action est effectuée dans le jeu.

• Déroulement du jeu

```
thomas@PC-DE-THOMAS:~/Documents/Cours_uB/Semestre_5/Systemes_Et_Reseaux/Projet$ make
gcc -c gestionnaireDeJeu.c
gcc -c joueurRobot.c
gcc -c jeu.c
gcc -c Carte.c
gcc -c gestionnaireDeStats.c
gcc -o gj gestionnaireDeJeu.o joueurRobot.o jeu.o Carte.o gestionnaireDeStats.o
gcc -c joueurHumain.c
gcc -o jh joueurHumain.o Carte.o
thomas@PC-DE-THOMAS:~/Documents/Cours_uB/Semestre_5/Systemes_Et_Reseaux/Projet$
```

En premier temps, nous lançons le Makefile pour compiler les fichiers.

```
thomas@PC-DE-THOMAS:~/Documents/Cours_uB/Semestre_5/Systemes_Et_Reseaux/Projet$ ./gj
[MJ]: Socket du maitre du jeu créé
[MJ]: En écoute sur le port 8087
Veuillez choisir le nombre de cnxJoueurH. Le nombre de cnxJoueurH : min 1 et max 10
1
Veuillez choisir le nombre des bots :
0
[MJ] En attente de la connexion des joueurs pour commencer.
[MJ] En attente du joueur n°1
```

Ensuite, nous lançons l'exécutable *gestionnaire de jeu*. Ce dernier nous propose donc d'entrer le nombre de joueurs que nous souhaitons et le nombre de robots. Juste après, il se met en écoute et attend qu'un joueur se connecte.

BELASSEL Meryem
NICOLLE Thomas

```
thomas@PC-DE-THOMAS:~/Documents/Cours_uB/Semestre_5/Systemes_Et_Reseaux/Projet$ ./jh
Veuillez choisir l'IP du serveur:
hostname
Veuillez choisir le port du serveur:
8087
Connexion réussite, attente réponse serveur...
╔═══════════════════════════════════════════════════════════════════════╗
|               .::Bienvenue dans le jeu 6 Qui Prend! ::.               |
╠═══════════════════════════════════════════════════════════════════════╣
|Principe du jeu: Les cartes de « 6 qui prend » ont 2 valeurs : une valeur|
|numérique (de 1 à 104) qui indique leur future position                |
|dans le jeu, et une valeur de 1 à 7 « têtes de boeufs » ,              |
|qui correspond à des points de pénalité. Le but est                    |
|de récolter le moins possible de têtes de boeufs. Le                   |
|gagnant est celui qui en comptabilise le moins à la fin du jeu.        |
╚═══════════════════════════════════════════════════════════════════════╝
En attente des joueurs...
┌--------|PLATEAU|--------┐
|[n°2 , 1 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] |
|[n°10 , 3 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] |
|[n°63 , 1 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] |
|[n°81 , 1 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] |
└------------------------┘
 .::DEBUT DU TOUR 1::.
┌-------- PAQUET DES CARTES --------┐
|[n°4 , 1 boeuf] [n°9 , 1 boeuf] [n°14 , 1 boeuf] [n°24 , 1 boeuf] [n°32 , 1 boeuf] [n°46 , 1 boeuf] [n°47 , 1 boeuf] [n°75 , 2 boeuf] [n°88 , 5 boeuf] [n°104 , 1 boeuf]
└---------------------------------┘
 Choisissez une carte !
```

Du côté joueur, l'utilisateur doit fournir l'adresse IP et le port auquel il veut se connecter. Dans notre cas, le joueur choisit de se connecter à la machine elle-même, puis au port 8087.
Une fois connecté au gestionnaire du jeu, un menu s'affiche pour l'accueillir ainsi que lui expliquer les règles du jeu.
La partie ne se lance qu'après que tous les joueurs soient prêts. Après la connexion de tous les joueurs, le plateau du jeu s'affiche suivi par l'affichage de la main du joueur. On lui demandera par la suite d'insérer dans le tampon d'entrée l'indice de la carte que le joueur souhaite choisir.
Dans notre cas, le joueur voulait jouer sa carte qui se trouve à l'indice 1 qui correspond à la carte de valeur numérique "4" et de valeur de boeuf "1".
En parallèle, le robot fait des choix d'une façon aléatoire.

```
 Pour poser la carte, veuillez entrer la rangée :

1
Le score du joueur 0 : 0
Le score du joueur 1 : 0
```

Après que tous les joueurs ont déposé leurs cartes sur les rangées respectives, le gestionnaire envoie les scores de tous les joueurs à tout le monde.

Une fois la carte choisie par tous les joueurs, le maître du jeu lui demande de sélectionner une rangée où il veut placer la carte choisie. Dans le cas de plusieurs joueurs, le premier joueur qui peut sélectionner la rangée est celui qui a choisi de jouer la carte de valeur numérique la plus faible parmi les autres cartes choisies.

BELASSEL Meryem
NICOLLE Thomas

```
[MJ] Joueur n°1 accepté!
[MJ]: En attente de la connexion des robots pour commencer.
[MJ]: En attente du robot n°0
[+] Joueur 1 s'est connecté au Jeu !
[MJ] Le nombre des joueurs total est : 2, dont 1 humain(s) et 1 robot(s)
[MJ]: Prépare la partie...
[+] Robot 1 s'est connecté au Jeu !
[MJ]: Distribue les cartes...
sh: 2: Syntax error: Unterminated quoted string
[MJ]: Prépare le plateau...
[MJ]: Mets les cartes sur le plateau...
[MJ]: Attends le choix de cartes des joueurs...
[MJ] Le robot 1 a proposé la carte [val=83, 1 boeuf]
[MJ] Envoie plateau au joueur 1
La chaîne a été ajoutée au fichier stats.txt.
La chaîne a été ajoutée au fichier stats.txt.
[MJ] Envoie main au joueur 1
La chaîne a été ajoutée au fichier stats.txt.
La chaîne a été ajoutée au fichier stats.txt.
[MJ] Attente d'une réponse du joueur 1
sh: 2: Syntax error: Unterminated quoted string
[MJ] Le joueur 1 a proposé la carte [valNum=4, 1 boeuf]
La chaîne a été ajoutée au fichier stats.txt.
[MJ] Le joueur 1 joue en 1 sa carte
```

Du côté du gestionnaire du jeu “maître du jeu”, nous pouvons apercevoir les commentaires qui décrivent en quelque sorte le déroulement de la partie. Nous remarquons aussi que de l’autre côté, le fichier “stats.txt” enregistre ces lignes.

```
 .::DEBUT DU TOUR 2::.
┌-------- PAQUET DES CARTES --------┐
|[n°9 , 1 boeuf] [n°14 , 1 boeuf] [n°24 , 1 boeuf] [n°32 , 1 boeuf] [n°46 , 1 boeuf] [n°47 , 1 boeuf] [n°75 , 2 boeuf] [n°88 , 5 boeuf] [n°104 , 1 boeuf]
└-----------------------------------┘
 Choisissez une carte !
```

Le deuxième tour débute et ainsi de suite jusqu’à ce qu’un joueur aille 66 têtes de bœuf ou bien il joue ses 10 cartes.

```
 Pour poser la carte, veuillez entrer la rangée :

4
Le score du joueur 0 : 6
Le score du joueur 1 : 13
Manche terminée.
thomas@PC-DE-THOMAS:~/Documents/Cours_uB/Semestre_5/Systemes_Et_Reseaux/Projet$
```

```
 ┌--------|PLATEAU|--------┐
|[n°14 , 1 boeuf] [n°47 , 1 boeuf] [n°52 , 1 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] |
|[n°20 , 3 boeuf] [n°29 , 1 boeuf] [n°32 , 1 boeuf] [n°57 , 1 boeuf] [n°60 , 3 boeuf] |
|[n°63 , 1 boeuf] [n°75 , 2 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] [n°0 , 0 boeuf] |
|[n°81 , 1 boeuf] [n°83 , 1 boeuf] [n°88 , 5 boeuf] [n°89 , 1 boeuf] [n°104 , 1 boeuf] |
└------------------------┘

MaxTetes = 0 tour= 10
```

BELASSEL Meryem
NICOLLE Thomas

```
[MJ]: Prépare la partie...
[MJ]: Distribue les cartes...
[MJ]: Prépare le plateau...
[MJ]: Mets les cartes sur le plateau...
[MJ]: Attends le choix de cartes des joueurs...
[MJ] Envoie plateau au joueur 1
.::DEBUT DU TOUR %d::.
[MJ] Envoie main au joueur 1
Choisissez une carte !
[MJ] Le joueur 1 a proposé la carte [valNum=4, 1 boeuf]
Le joueur 1 va recevoir le message pose carte
[MJ] Le joueur 1 joue en 1 sa carte
[MJ] Le joueur 0 place sa carte à la rangée 1
MISE A JOUR DU PLATEAU :
MaxTetes = 0 tour= 1
[MJ] Le joueur 2 joue en 2 sa carte
[MJ]: Attends le choix de cartes des joueurs...
.::DEBUT DU TOUR %d::.
[MJ] Envoie main au joueur 1
Choisissez une carte !
[MJ] Le joueur 1 a proposé la carte [valNum=9, 1 boeuf]
[MJ] Le joueur 1 joue en 1 sa carte
Le joueur 1 va recevoir le message pose carte
[MJ] Le joueur 0 place sa carte à la rangée 1
MISE A JOUR DU PLATEAU :
MaxTetes = 0 tour= 2
[MJ] Le joueur 2 joue en 2 sa carte
[MJ]: Attends le choix de cartes des joueurs...
.::DEBUT DU TOUR %d::.
[MJ] Envoie main au joueur 1
Choisissez une carte !
[MJ] Le joueur 1 a proposé la carte [valNum=28, 1 boeuf]
```

Ceci représente le fichier texte qui contient les sorties redirigées par le gestionnaire de jeu.