Projet Puissance 4

(2008/2009)

Réseaux et Systèmes

Deuxième année

L'objectif du projet est d'écrire un système clients/serveur permettant de jouer à Puissance 4.

Évaluation de ce projet

Vous pouvez réaliser le travail par groupe de trois étudiants.

Vous devez rendre un mini-rapport de projet (5 pages maximum, format pdf). Vous y détaillerez les difficultés auxquelles vous avez été confronté, et comment vous les avez résolues. Vous indiquerez également le nombre d'heures passées sur les différentes étapes de ce projet (conception, codage, tests, rédaction du rapport) par chaque membre du groupe.

Vous devrez nous faire une démonstration de votre projet et être prêts à répondre à toutes les questions techniques sur le codage de l'application. Les soutenances auront lieu autour du 10 décembre.

Comment rendre votre projet

En plus du mini-rapport et de vos sources, vous devez fournir un Makefile dont la cible par défaut compilera votre projet sous forme de deux binaires : `serveur` (qui n'attend pas d'argument en ligne de commande) et `client` (qui attend l'adresse et le port du serveur en arguments).

Un projet ne compilant pas correctement sera sanctionnée par une note adéquate.

Pour rendre votre copie, vous devez placer les fichiers nécessaires dans un répertoire sur neptune, vous placer dedans, et invoquer la commande suivante (seuls les fichiers sources (et le pdf) sont copiés). La tricherie sera **sévèrement punie**. Voir `http://www.loria.fr/~quinson/teaching.html#triche`

`/home/EqPedag/quinson/bin/rendre_projet RS`

Avant le lundi 8 Décembre, à 23h59.

(le script n'acceptera pas de soumission en retard)

Voir `http://www.loria.fr/~quinson/teach-RS.html` d'éventuelles informations complémentaires.

1 Règle du jeu

Le jeu est constitué d'une grille composée de 7 colonnes et 6 lignes. Il se joue à 2 joueurs. Chaque joueur dispose de 21 pions d'une couleur (rouge ou noire). Dans les versions développées, vous pouvez, si vous le souhaitez, remplacer la couleur rouge par un R et la couleur noire par un N.

Il faut tirer au sort pour connaître le joueur qui commence la partie. Chaque joueur, à tour de rôle, doit insérer un pion dans l'une des colonnes de la grille. Le pion descend jusqu'à ce qu'il atteigne la dernière case libre (càd, il s'agit soit de la case du bas, ou la case du dessous est déjà occupée).

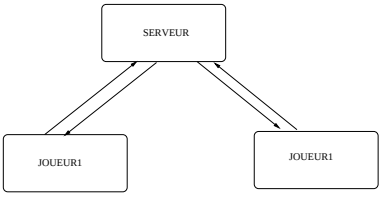
Il faut continuer ainsi jusqu'à soit :

- l'abandon de la partie par un joueur (l'autre est déclaré vainqueur) ;
- l'un des joueurs aligne 4 pions horizontalement, verticalement ou en diagonal (il est alors déclaré vainqueur) ;
- Il n'y a plus de jetons de disponibles (la partie est déclarée nulle).

À la fin de la partie, soit les joueurs décident d'arrêter de jouer soit ils recommencent une nouvelle partie.

2 Architecture

Les parties sont gérées de manière centralisée par un serveur qui reçoit les demandes de positionnement des pions par les joueurs et leur transmet les résultats de leurs décisions.



3 Protocole

Les messages échangés entre un joueur et le serveur sont les suivants. *Il est très important que votre implémentation respecte scrupuleusement la syntaxe des messages échangés, car nous testerons votre serveur avec des clients que nous aurons réalisé.*

3.1 Messages de connexion au jeu

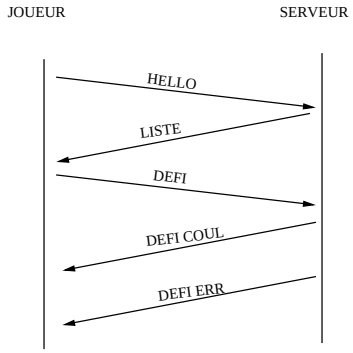
HELLO <pseudo>
Permet de se déclarer au niveau du serveur avec un pseudo.

LISTE <nbre joueurs> <pseudo1> ... <pseudoN>
Le serveur retourne la liste des joueurs potentiels actuellement connectés au serveur.

DEFI <pseudo>
Le joueur défie le joueur <pseudo> (qui ne peut décliner le défi).

DEFI COUL <N/R> <N/R>
Suite au défi, le serveur donne la couleur attribuée au joueur (**n**oire ou **r**ouge), ainsi que la couleur qui commence la partie.

DEFI ERR <msg>
Suite au défi, le serveur peut renvoyer un message d'erreur avec un message associé.



3.2 Messages relatifs au jeu

COUP COL <N>
Le joueur précise la colonne où il veut déposer le pion.

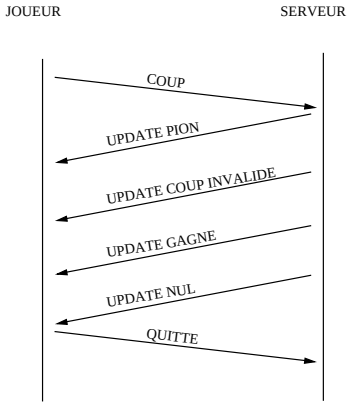
UPDATE PION <N/R> <X> <Y>
Le serveur envoie aux deux joueurs le résultat qui fait suite au coup réalisé par le joueur dont la couleur est <N/R>. Les valeurs <X> et <Y> précisent le numéro de ligne et de colonne où le pion a été positionné dans la grille.

UPDATE COUP INVALIDE
Le serveur indique au joueur que le coup est invalide.

UPDATE GAGNE <N/R> <X> <Y> <dir>
Le serveur envoie aux deux joueurs que le joueur de couleur <N/R> a gagné. <X> et <Y> et <dir> précisent la position de l'alignement de quatre pions créé (dans le cas où plusieurs alignements sont créés par le coup gagnant, l'un des alignements est choisi arbitrairement par le serveur). Voir aussi figure 1.

UPDATE NUL La partie s'est soldée par un match nul.

QUITTE Le joueur abandonne la partie.

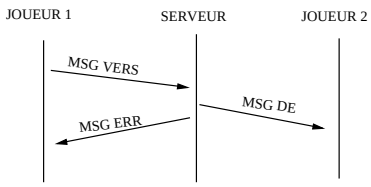


3.3 Messages du forum de discussion

MSG VERS <pseudo> <blabla>
Un message est envoyé, via le serveur, au joueur <pseudo>.

MSG DE <pseudo> <blabla>
Un message est envoyé, via le serveur, par le joueur <pseudo>.

MSG ERR <msg>
Le serveur renvoie un message d'erreur avec un message associé.



4 Ce que vous devez implémenter

Il est conseillé d'écrire plusieurs versions du client et du serveur, en ajoutant les fonctionnalités au fur et à mesure. Pensez à sauvegarder en lieu sûr le code de l'une des versions avant de passer à la suivante afin de vous assurer de pouvoir montrer quelque chose de fonctionnel lors de la soutenance.

Voici un enchainement possible :

- **Version 0 :** tout est réalisé dans un même processus avec un découpage en différentes fonctions (pas besoin de rendre cette version) ;
- **Version 1 :** vous découpez en plusieurs processus : un processus par joueur et un processus pour le serveur. La communication entre les différents processus devra se faire par tube nommé (cf. l'appel système `mkfifo`).
- **Version 2 :** la communication peut être est réalisée via des sockets réseaux quand l'option `-net` est passée aux programmes ;
- **Version 3 :** le serveur peut gérer plusieurs parties à la fois. L'utilisation de threads est demandée (un thread par partie en cours) ;
- **Version 4 :** vous ajoutez la fonctionnalité de discussion entre deux joueurs ; de ce fait le client doit être à même de gérer différents threads (un pour gérer les interactions avec le joueur, et un pour attendre les messages venus du serveur). Vous devez passer par le serveur car lui seul connaît l'adresse du "pseudo" de l'autre joueur.

5 Ce qu'il n'est pas nécessaire d'implémenter

La correction ne tiendra aucun compte de l'interface graphique. Vous pouvez réaliser une interface graphique si vous le souhaitez, mais uniquement une fois que les 5 versions du projet fonctionnent. Avant cela, un affichage texte (comme dans la figure 2) suffit amplement.

1	horizontal (vers la droite)
2	vertical (vers le bas)
3	diagonal descendant vers la droite
4	diagonal descendant vers la gauche

FIG. 1 – Signification de <dir> dans un message UPDATE GAGNE.

```
.......
.......
N..R...
R..N...
N..N...
RNRR...
0123456
```

FIG. 2 – Exemple d'affichage attendu.