UFR ST Besançon 2015-2016
Master 1 Informatique

Systèmes Communicants et Synchronisés
Projet - Jeu Tic-Tac-Toe en réseau

---

1 Description du projet

Le but de ce projet est l'implantation de la communication permettant de jouer en réseau au jeu Tic-Tac-Toe. Les règles du jeu sont données dans l'énoncé du projet MOIA. La suite définit les spécifications du travail à réaliser pour la partie communication. Le travail est à remettre en binôme.

Chaque joueur est représenté par un processus joueur écrit **en langage C**. L'intelligence du jeu Tic-Tac-Toe (appelée moteur IA) est développée en Prolog et est utilisée dans un **programme Java** qui communique avec le processus joueur, écrit en C. Les parties se jouent à deux joueurs et sont administrées par un processus serveur, arbitre du jeu, écrit **en langage C**. Les processus joueurs communiquent, à travers des sockets, avec le serveur.

1.1 Le joueur

Voici le comportement d'un processus joueur :

1. Il envoie au serveur une requête **PARTIE** pour lui demander de jouer, en précisant son nom. En réponse, le serveur lui donne le nom de son adversaire et lui indique son symbol. Le joueur ayant les croix commencera la partie.
2. Lorsque c'est à son tour de jouer, le joueur consulte son moteur IA pour connaître le coup à jouer. Il constitue une requête **COUP** qu'il envoie au serveur, puis il en attend la validation.
3. Il attend ensuite le coup de son adversaire ainsi que sa validation. Ce temps peut être utilisé par le joueur pour pré-calculer les prochains coups.
4. Si la partie n'est pas finie, le joueur informe son moteur IA du coup joué par l'adversaire et retourne en 2.

1.2 Le serveur

Le processus serveur réalise les actions suivantes :

1. Il attend d'avoir deux requêtes **PARTIE** de la part de joueurs pour répondre à chacun en donnant les croix au joueur dont la requête est reçue en premier.
2. Il attend ensuite alternativement une requête **COUP** du joueur ayant les croix puis du joueur ayant les ronds. Cette attente doit être bornée dans le temps par une constante **TIME_MAX** fixée à 6 secondes. Pour chacune des requêtes reçues, il l'envoie à l'adversaire. Ensuite, il vérifie la validité de ce coup (voir 1.4 - une bibliothèque vous sera fournie) et informe chacun des joueurs si le coup est valide ou non. Cette réponse intègre également l'état de la partie (si elle doit continuer ou non).
3. Si le coup d'un joueur ne termine pas la partie (**GAGNANT**, **NULLE**, **PERDU**) il recommence en 2.

Une partie est finie si un des joueurs gagne ou perd (coup validé par le serveur), s'il y a partie nulle ou si un des joueurs joue un coup invalide.

1.3 Protocole du serveur

Toutes les communications se feront en mode connecté. Vous trouverez sous moodle dans le module SCS - rubrique Projet, le fichier `protocoleTicTacToe.h` qui définit le protocole d'accès au serveur de jeu Tic-Tac-Toe.

Voici les règles retenues pour l'utilisation de ces types :

PARTIE : cette requête permet à un joueur de demander une partie. Le nom à donner est le login LDAP du joueur. Il obtient en retour le nom de son adversaire. Si le symbole reçu de la part du serveur est `CROIX`, il commence à jouer. Sinon il attend de recevoir le premier coup de son adversaire. En fin de partie, il doit se déconnecter et finir son exécution.

COUP : une requête `TypCoupReq` est envoyée par le joueur au serveur pour jouer un coup. Cette requête donne le symbol du joueur `symbolJ`, la position du symbole joué `pos` et le nombre de sous-plateaux gagnés suite au placement du symbole `nbSousPlatG`. Lorsque le joueur place son symbole, il doit initialiser, dans la structure `pos`, le numéro du plateau joué (`numPlat`) et le numéro du sous-plateau joué (`numSousPlat`). Ces numéros sont des entiers définis par les énumérations `TypPlat`, respectivement `TypSousPlat`.

En réponse à la requête, le serveur retourne une validation du coup `TypCoupRep` avec une valeur d'erreur initialisée à `ERR_OK` si il n'y a pas eu d'erreur ; le champ `validCoup` précise si le coup est valide, en timeout ou s'il y a triche et le champ `propCoup` précise si le coup termine la partie en gagnant (`GAGNANT`), en perdant (`PERDU`) ou en match nul (`NULLE`), ou si la partie peut continuer (`CONT`).

1.4 Validation des coups

Quelques précautions sont prises pour éviter les erreurs répétées, voire moins bien intentionnées.

— Si un coup est erroné (le symbole n'est pas placé correctement sur la grille de jeu), le serveur retourne une valeur `TRICHE` dans le champ `validCoup`.

— La recherche dans le moteur IA est limitée dans le temps. Le temps d'attente est calculé à partir de l'instant où le serveur envoie la réponse au coup joué par un joueur. Si la limite est dépassée, le coup est considéré comme nul et le serveur envoie au joueur et à l'adversaire un message `TypCoupRep` avec une valeur `TIMEOUT` dans le champ `validCoup` (l'adversaire ne reçoit pas le coup joué). Dans ce cas, la partie est perdue par le joueur. Attention, un joueur peut donc recevoir un `TIMEOUT` alors qu'il est en train de jouer.

Dans ces deux situations, le coup est considéré invalide par le serveur.

2 Remise du travail

Pour la remise de ce projet nous attendons le code source des programmes serveur et client et un rapport donnant des explications sur votre travail.

2.1 Développements

Vous devez développer le joueur (ainsi que son moteur IA) et le serveur sur la base du protocole de communication décrit ci-dessus. Le développement SCS sera réalisé en binôme selon la répartition suivante : le joueur et le programme Java appelant le moteur IA, d'une part et le serveur de jeu, d'autre part.

Les messages seront échangés sur des sockets.

Vous nous fournirez les versions de votre joueur et de votre serveur (**incluant la partie MOIA**) en le déposant comme devoir dans le module SCS sur moodle, pour le **6 mai à 23h00**, au plus tard. Le fichier, portant les deux noms du binôme, sera une archive .tar.gz qui contiendra un répertoire, avec le même nom. Dans ce répertoire, mettre tous les fichiers de votre projet (sauf les fichiers temporaires .o, ~, etc.) et un fichier appelé `joueurTicTacToe` (exécutable, script, etc.).

Remarque sur la programmation : pour que vos programmes soient facilement lisibles, vous respecterez le standard de programmation C/Java.

2.2 Rapport

Pour expliquer votre réalisation, vous nous remettrez un rapport (5-6 pages) qui contiendra obligatoirement les informations suivantes :

— l'utilisation du protocole de communication avec le serveur,

— la mise en place d'un protocole avec le moteur IA,

— les explications sur le code, la structure du joueur et du serveur.

La longueur de ce rapport n'est pas un facteur déterminant, portez plus d'attention à sa clarté et à la structuration de vos explications. Ce rapport est à remettre en même temps avec les sources du projet, au format pdf.

3 Championnat

Les joueurs s'affronteront en championnat (le mercredi 11 mai), en utilisant une version exécutable du serveur que nous vous mettrons à disposition.

Pour faciliter le déroulement de ce championnat vous devez respecter le mode de lancement suivant : le joueur prend en paramètre le nom de la machine et le numéro de port du processus serveur pour s'y connecter, ainsi que le nom le désignant dans le jeu. La forme exigée est la suivante :

```
./joueurTicTacToe hostServeur portServeur nomJoueur [opt]
```

où
`hostServeur` et `portServeur` - désignent le nom de la machine et le port du serveur,
`nomJoueur` - désigne le nom du joueur,
`[opt]` - paramètres supplémentaires qui peuvent être définis.

Il faut impérativement fournir un fichier README contenant la ligne de commande avec la sémantique des paramètres.

Attention. Tout manquement à ces règles peut vous exclure du championnat mais, si vous vous y prenez un peu à l'avance, il sera possible de nous demander une validation (lors de la dernière séance de TP).