Licence 2 - Info 4B

Semestre 4 - Année 2022

Projet Principes des Systèmes d'Exploitation • à rendre la semaine du 11 Avril

Cadre général du projet

Le projet à pour but d'appliquer les principes des systèmes d'exploitation afin de réaliser l'une des 3 applications décrites dans la suite du document : 1) Exploitation des données de LiChess, 2) un jeu des années 80 étendu avec un mode multi-joueurs et un mode réseau, 3) le jeu de Core War au cours duquel des programmes s'affrontent pour conquérir la mémoire.

Contraintes pour la réalisation

Le projet est à réaliser par groupe de maximum deux étudiants. La programmation se fera en utilisant la plate-forme Java. Il n'est pas obligatoire de développer une interface graphique. Votre programme devra être conçu de façon modulaire (packages, interfaces, classes respectant une encapsulation stricte, classes utilitaires, constantes, fichiers de configuration, etc.) et il devra refléter la conception en couches d'un système d'exploitation (les couches et leurs fonctionnalités sont à détailler dans le rapport). Les structures de données que vous utiliserez ne sont pas nécessairement dynamiques (structure à base de références), vous pouvez utiliser des tableaux, cependant les collections proposées par Java seront d'une grande utilité. L'emploi des membres de classe déclarés *static* devra être justifié. Il est également très fortement conseillé de respecter les conventions d'écriture de programmes Java qui sont des règles pour faciliter la compréhension et la maintenance du code (`https://www.jmdoudoux.fr/java/dej/chap-normes-dev.htm`). La lisibilité du code sera évaluée.

Vous devez faire une présentation de votre projet avant les vacances de printemps (semaine du 11 avril 2022). Ce jour là vous remettrez un rapport papier. Pour ce document, les consignes **strictes** de présentation sont : au moins 15 pages, police 12pts maximum, marges 2cm maximum. Le rapport doit comporter au moins les éléments suivants :

- **une analyse fonctionnelle** du sujet, précisant, entre autre, toutes les règles de fonctionnement de votre application et un découpage du problème en sous-problèmes (pour aboutir aux grandes lignes de l'architecture logicielle), c'est-à-dire les classes les plus importantes et leurs liens ;
- **une description des structures de données** envisagées et retenues ;
- **la spécification des classes principales** et des méthodes essentielles ;
- **l'architecture logicielle détaillée** de votre application, c'est-à-dire, dans le cadre du module Info4B, une conception en couches fonctionnelles à l'image de ce qui est réalisé dans l'architecture d'un système d'exploitation. Pour chaque couche vous spécifierez les classes qui implémentent les services de la couche fonctionnelle ;
- une description des **algorithmes principaux** ;
- **un jeu de test** montrant que votre programme fonctionne.

Il est conseillé de rédiger le rapport en utilisant LaTeX car il permet d'inclure facilement des extraits de code source avec une mise en évidence des éléments syntaxiques. On trouve des applications permettant de rédiger collaborativement des documents LaTeX (`https://fr.sharelatex.com/`, `https://fr.overleaf.com/` ou sur le serveur `iemgit` accessible par VPN en demandant au préalable l'activation d'un compte). Pour le partage de code entre binôme l'outil git et son interface Web GitHub sont recommandés. Dans le document, il est conseillé d'utiliser des schémas pour illustrer vos propos. Des diagrammes UML comme le diagramme de classes ou de séquences peuvent être ajoutés à votre rapport.

Une archive[1] (tgz, jar ou zip exclusivement) de l'ensemble de code source et de l'exécutable de votre programme devra être produite à la date du jour de la démonstration, puis déposée sur Plubel dans l'espace du cours prévu à cet effet.

1. Les autres formats d'archive ne sont pas autorisés

L'évaluation se fait sur la base du rapport papier, de la qualité technique de la réalisation, du déroulement de la démonstration. Lors de la démonstration vous devrez avoir préparé un scénario et des jeux de tests afin de présenter les fonctionnalités de votre application (vous disposerez de 10 minutes).

Le descriptif des projets est volontairement synthétique. Vous devez/pouvez déposer des questions dans le forum Discord prévu à cet effet. Il faut également exploiter les références données.

Sujet 1 : Exploitation des données de LiChess

L'objectif est d'exploiter plusieurs machines pour traiter un jeu de données de grande taille, d'effectuer des recherches simples et d'appliquer un algorithme plus sophistiqué sur l'ensemble des données.

Vous pouvez télécharger les données des parties d'échecs à l'URL suivante : https://database.lichess.org/ et ensuite développer un système pour les exploiter à partir de différents programmes.

Le système sera composé d'un serveur chargé de réceptionner les demandes venant de clients (qui peuvent être localisés sur des machines distantes). Le serveur délègue le traitement des demandes à des threads afin d'accélérer la production de résultats grâce à la parallélisation (une demande est déléguée à plusieurs threads).

Le jeu de données peut être découpé en plusieurs parties. L'utilisation des Hashtables pour prétraiter les demandes est essentielle. Elles peuvent servir à stocker des informations résumées pour ensuite guider la récupération des données correspondantes.

Les clients peuvent consulter par exemple :

- une partie spécifique et la visualiser pas à pas ;
- trouver toutes les parties d'un joueur ;
- les 5 ouvertures les plus jouées (ne prendre en compte que le premier coup) ;
- les parties les plus courtes ;
- lister les joueurs les plus actifs, les plus actifs sur une semaine, etc. ;
- calculer le joueur le plus fort au sens du PageRank ;
- le plus grand nombre de coups consécutifs cc qui soient communs à p parties (en cherchant à maximiser cc et p, l'ensemble des coups consécutifs peut représenter un début ou une fin de partie) ;
- etc.

À partir d'un client on envoie une (ou plusieurs) requête au serveur qui peut traiter plusieurs demandes de clients à la fois en allouant n threads pour les calculs. Cependant, le serveur ne soit pas se surcharger et fixe $n < MAX$ avec $MAX = 4$ par exemple. Les autres demandes sont mises en attente.

Pour l'algorithme PageRank, vous devrez implanter une version itérative (la version utilisant valeurs propres et vecteurs propres n'est pas demandée).

Références sur l'algorithme PageRank :

- https://fr.wikipedia.org/wiki/PageRank

Sujet 2 : Lode Runner

Lode Runner est un jeu de plates-formes développé par Douglas Smith et publié par Brøderbund en 1983. Le joueur évolue dans un décor (tableau) constitué d'échelles, de murs de briques et de passerelles. Le but du joueur est de ramasser des éléments (paquets, lingots, etc.) qui ont été disséminés dans le tableau. Une fois que tous les éléments sont collectés le joueur doit sortir du tableau par l'échelle la plus haute.

Des adversaires (automatiques ou humains) essayent d'attraper le joueur, celui-ci peut creuser les briques autour de lui pour ensevelir ses adversaires. Il peut également creuser les briques pour accéder à une portion du tableau. Les trous ainsi créés se rebouchent en quelques secondes. Les adversaires ensevelis réapparaissent en haut du tableau. Si un joueur est coincé ou touché par un adversaire, il perd une vie (sur les cinq prévues en début de partie) et réapparaît en haut du tableau.

Vous avez libre choix d'adapter ou d'ajouter des règles en respectant l'esprit du jeu. Vous devez implanter un mode collaboratif (avec plusieurs joueurs qui collectent les éléments) et un mode combat (pour lequel les adversaires peuvent être contrôlés par d'autres humains sur d'autres machines).

Vous devez impérativement réaliser les fonctionnalités suivantes :

- les adversaires doivent pourvoir être contrôlés soit par des joueurs (mode combat) ou soit par le programme (IA) ;
- il doit être possible de jouer en réseau à plusieurs (mode coopératif) avec plusieurs machines dont l'une est serveur ;
- les scores doivent être enregistrés sur un serveur et la liste des meilleurs joueurs doit être maintenue à jour, il doit être aussi possible de définir des équipes en mode collaboratif ;
- le chargement de tableaux de jeu (niveaux) à partir de leur définition sous la forme de fichiers texte.

Avant d'implanter le réseau vous devez avoir réalisé et testé le fonctionnement du jeu en mode machine unique. Il n'est pas demandé d'interface graphique, vous pouvez utiliser les caractères ASCII pour représenter les éléments du décor.

Références :

- http://fr.wikipedia.org/wiki/Lode_Runner
- http://en.wikipedia.org/wiki/Lode_Runner donne plus de détails
- https://archive.org/details/msdos_Lode_Runner_1983_1983 jouable en ligne

Sujet 3 : Core War

On se propose d'appliquer les concepts des systèmes d'exploitation afin de réaliser un simulateur *Mars* dont la description se trouve dans l'article de A. K. Dewdney cité dans les références et dont la traduction en français sera déposée sur la plateforme Plubel. Votre simulateur doit être capable d'exécuter simultanément plusieurs programmes écrits en *RedCode*. La taille mémoire de la machine est fixée à l'initialisation du jeu. Le programme doit déclarer un joueur gagnant et classer le ou les autres joueurs.

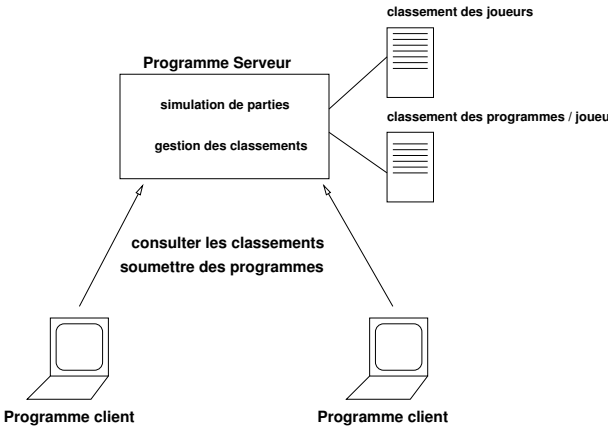

FIGURE 1 – Jeu de Core War en réseau

Les programmes *Mars* sont des fichiers texte que l'on charge en début de partie, qui sont interprétés lors de l'exécution (voir la description de la syntaxe dans les références).

En plus des éléments décrits dans les références, les fonctionnalités principales que vous avez à ajouter sont de deux types :

- par rapport à l'article original de Dewdney vous devez permettre la soumission de programmes par le réseau, c'est-à-dire développer une partie serveur et une partie client. Un même serveur peut héberger plusieurs parties se déroulant en même temps (figure 1). Une partie démarre lorsque chaque joueur participant a envoyé son programme ;

— par rapport à la gestion des joueurs et des programmes : à mesure des parties gagnées un joueur progresse dans le classement (stocké sur la machine qui héberge le programme serveur). On gardera aussi le *hit parade* des noms des programmes gagnants. Avant de lancer une partie, il est possible de consulter les classements.

Si vous souhaitez réaliser d'autres extensions du langage *RedCode*, votre interpréteur devra être capable de supporter deux syntaxes : celle des programmes en *RedCode Base* correspondant à la description du langage dans l'article de A. Dewdney et la syntaxe du *RedCode Étendu* correspondant à vos ajouts. On doit pouvoir sélectionner l'une des syntaxes au lancement des programmes.

Références :

— Article original en anglais : https://www.corewars.org/sciam/
— http://www.koth.org/index.html
— https://fr.wikipedia.org/wiki/Core_War