

Simulation d'un système boursier

Mini projet IN301/IN3ST01 - 2011-2012

Sujet rédigé par Jean Cousty et Laurent Najman,

très librement inspiré d'un projet de Jérôme Gueydan pour l'ENSTA

8 décembre 2011

Table des matières

1	Introduction	4
1.1	Le principe des bourses EuroPrevious	5
1.2	Le principe des ordres de transaction	5
1.3	Le central des salles de marchés	6
1.4	Le routage	7
2	Cahier des charges	7
2.1	Cahier des charges fonctionnelles	8
2.1.1	Architecture fonctionnelle	8
2.1.2	Terminal	8
2.1.3	Serveur d'acquisition	9
2.1.4	Serveur d'exécution	9
2.1.5	Le réseau inter-boursier	10
2.2	Cahier des charges techniques	10
2.2.1	Contraintes générales	10
2.2.2	Mémoire des serveurs	10
2.2.3	Nombre de processus	10
2.2.4	Paramètres	11
2.2.5	Gestion des échanges par tuyaux	11
2.3	Comment tester sans s'y perdre?	11
2.4	Livrables	12
3	Conduite du projet	12
3.1	Introduction	12
3.1.1	Un projet en plusieurs étapes	12
3.1.2	Méthode "diviser pour régner"	13
3.2	Étape 1 : les fonctions de communication	13

3.3	Étape 2 : réalisation de programmes indépendants	14
3.3.1	Processus : <i>Terminal</i>	14
3.3.2	Processus : <i>Execution</i>	14
3.3.3	Processus : <i>Acquisition</i>	14
3.4	Étape 3 : création d'un réseau Bourse	15
3.4.1	Préparation de la communication par tuyaux	15
3.4.2	Raccordement	15
3.5	Étape 4 : création du réseau inter-boursier	15
3.5.1	Processus : <i>Inter-boursier</i>	15
3.5.2	Raccordement	16
4	Évolutions complémentaires et optionnelles	16
4.1	Ouverture et clôture de séances	16
4.2	Bilan des entreprises cotées en bourse	16
4.3	Utilisation de <code>socket</code>	16
5	Annexes	17
5.1	Codes disponibles en ligne	17
5.2	De l'intérêt des standards pour assurer une meilleure interopéra- bilité	17
5.3	Protocoles de communication et format de messages	18
5.4	Cotation des actions en bourse	18
5.5	Redirection des entrées et des sorties	19

Avertissement

Le projet suivant est à faire par groupe de deux étudiants (en une séance de TD de 2h, deux séances de TP de 3h, 12h de projet encadré, plus un **petit** travail personnel si nécessaire), et à rendre pour janvier 2012, à une date qui sera précisée ultérieurement.

Pour faire court, sachez que tous les projets “semblables” auront une note de zéro, et que les auteurs seront convoqués en conseil de discipline. Ceux qui arriveront à faire la preuve qu'ils ont écrit le projet duquel les autres se sont inspirés n'auront aucune pénalité supplémentaire par rapport à la note de zéro. Si vous faites le projet à plus de deux, un coefficient de réduction proportionnel sera appliqué. Par exemple, si vous faites le projet à trois, la note sera réduite d'un tiers. Aucun bonus ne sera donné pour un étudiant travaillant seul.

Plagiat

Le plagiat est l'utilisation, sans citation appropriée, du travail intellectuel d'une autre personne dans un travail soumis à évaluation. Ce qui suit est très fortement inspiré et traduit de la page

<http://www.csd.abdn.ac.uk/teaching/handbook/both/info.php?filename=cheating.txt>.

Quand vous écrivez un rapport ou du code qui contient des parties ou qui paraphrase le travail d'autres personnes, vous devez clairement le signaler. En particulier, les citations doivent être données entre guillemets, avec les références appropriées.

1. Le code soumis pour évaluation doit clairement être annoté avec le nom de l'étudiant qui a soumis l'exercice, avec la date de rédaction.
2. Quand un exercice contient du code qui n'a pas été écrit par l'étudiant qui soumet le travail, ou contient du code écrit par l'étudiant lui-même à un autre moment, le code en question doit être clairement identifié et annoté avec le nom de l'auteur, le copyright (si différent), la date d'achèvement ou de publication, et une référence à la source (par exemple une URL ou une référence bibliographique).
3. En principe, la discussion avec d'autres étudiants du contenu du cours et des exercices non-évalués est encouragée, car une telle discussion amène généralement à une meilleure compréhension du sujet. Cependant, les discussions doivent rester à un niveau général, et ne doivent pas descendre à un niveau détaillé de conception ou de codage. Le travail soumis **pour évaluation doit être un travail individuel**. Si vous soumettez un travail produit conjointement avec une autre personne, ou qui inclut le travail de quelqu'un d'autre, **ceci doit être clairement indiqué, et le code en question doit être clairement identifié**. L'évaluation du travail se fera sur la base de la valeur ajoutée par l'étudiant.
4. De manière identique, bien que nous encourageons la réutilisation de logiciel existant comme une bonne pratique d'Ingénierie Logicielle, les origines d'une telle ré-utilisation doivent **être clairement indiquées, et le code identifié**. L'évaluation du travail se fera sur la base de la valeur ajoutée par l'étudiant.
5. Quand le travail est à réaliser en groupe de deux étudiants, le rapport écrit ou le travail doit clairement identifier et distinguer ce qui a été réalisé par un des étudiants, ou ce qui a été fait par le groupe en entier.

Consignes

- Vous devez m'envoyer avant janvier 2012 dernier délai (la date effective sera précisée ultérieurement) une archive comprimée en tar.gz, contenant le code, le makefile, et les rapports du projet sous format électronique. Les rapports papiers doivent être déposés le même jour au secrétariat du Département Informatique. Un barème de "moins deux points" par jour de retard sera appliqué. **Une archive qui ne sera ni du tar.gz ni du .zip ne sera pas corrigée.**
- Le projet devra se compiler en exécutant la commande make sur une machine tournant Linux. **Un projet qui ne respectera pas cette consigne ne pourra pas être corrigé.**
- Le rapport du projet décrira les diverses solutions aux différents problèmes rencontrés, et **justifiera** la solution choisie.

- Votre note dépendra de la qualité du rapport, et de la qualité du code.
- Le barème prend en compte pour une part la rédaction de la réponse, et également l'écriture du code correspondant aux choix que vous aurez fait. Vous pourrez donc avoir des points si vous n'avez pas programmé vos propositions, mais vous n'aurez pas forcément tous les points si le code que vous avez écrit n'est pas proprement justifié.

Les consignes de remises du projet seront précisées sur la page web du cours. D'autres éléments, comme le barème d'évaluation, y seront également notés.

Remarque

L'énoncé peut sembler épais de prime abord. Ce n'est pas le cas : il se résume aux pages 4 à 11. Le reste du document a pour objectif de vous aider dans la réalisation du projet, en indiquant des éléments de solutions, en attirant l'attention sur les points essentiels, et en donnant une démarche saine de progression.

La rédaction du texte du sujet est inspiré par un projet anciennement proposé par Jérôme Gueydan pour l'ENSTA et bénéficie d'une expérience de plusieurs années à l'ESIEE.

1 Introduction



Nous sommes le 11 septembre 2012. Environ trente mois après un premier plan de sauvetage, la Grèce puis, à sa suite, par un effet de domino et l'influence des avis émis par les agences de notations, le Portugal, l'Espagne, l'Italie, la France, l'Allemagne, l'Europe, les Etats-Unis et la Chine ont fait banqueroute anéantissant le système boursier d'échange d'actions que nous connaissions jus-

qu'à présent. La communauté européenne décide de refonder son système financier. Elle confie cette tâche à un consortium, nommé EuroPrevious, composé d'experts financiers spécialisés dans les solutions d'avenir. Afin de réaliser le système informatique responsable de la gestion des échanges d'actions, EuroPrevious lance un appel d'offre auquel vous répondez. Pour cela, vous devez concevoir une simulation du système informatique qui permettra d'interconnecter les différentes bourses européennes.

Voici une description succincte de ce système. Un certain nombre de terminaux est disposé dans chaque salle de marché. Les commerçants¹ se connectent aux terminaux pour passer leurs ordres de transactions. Un commerçant peut acheter ou vendre des actions cotés aussi bien dans sa propre salle de marché que dans n'importe quelle autre du groupe EuroPrevious. Lorsqu'un commerçant passe un ordre, il souhaite recevoir un accusé de réception l'informant si cet ordre a pu être réalisé ou non. Dans le cas positif, il souhaite également savoir combien lui a coûté/rapporté l'achat/la vente de ses actions. L'objectif du

1. Également appelés *traders*, merci la loi Toubon !

projet est de simuler ces échanges d'ordres de transactions entre les terminaux des commerçants et les serveurs centraux des différentes bourses. Vous recevez le cahier des charges ci-dessous.

Avant de présenter le sujet, examinons le fonctionnement des bourses EuroPrevious et des ordres de transactions.

1.1 Le principe des bourses EuroPrevious

Dans le système financier mis en place par EuroPrevious, chaque bourse fonctionne comme un magasin.

Elle dispose d'un **stock** d'actions qu'elle vend et achète aux commerçants.

La **cotation** d'une action dépend de la **valeur intrinsèque** de l'entreprise et du nombre d'actions de ce type actuellement stockées par la bourse. Plus le stock est grand, plus le prix de l'action est faible. Plus le stock est petit, plus le prix de l'action est élevée.

Nous supposons également que la **capacité** de stockage de la bourse est globalement limitée, c'est à dire qu'à un instant donné elle ne peut avoir plus d'un certain nombre d'actions en stock. Ce système introduit un moyen de protection contre la panique. Pour s'en convaincre, supposez que tous les commerçants, pris de panique, souhaitent se débarrasser au même instant de toutes leurs actions. Au bout d'un certain nombre de transactions, le stock de la bourse atteint sa limite : les commerçants ne peuvent plus lui vendre d'actions. Pour débloquer la situation et pouvoir de nouveau vendre des actions, les commerçants doivent donc d'abord racheter des actions à la bourse. L'offre et la demande se trouvent ainsi régulées.

1.2 Le principe des ordres de transaction

Le principe des ordres de transaction met en relation plusieurs acteurs :

- le commerçant, qui souhaite envoyer un ordre de transaction ;
- le terminal informatique de la salle de marché où le commerçant se connecte ;
- le central de la salle de marché (par exemple, la bourse de Paris) à laquelle est connecté le terminal ;
- le central de la salle de marché à qui l'ordre de transaction est destiné.

Les terminaux des salles de marché sont reliés à leur central respectif par la toute dernière technologie 25Gmax de réseau sécurisé. Les différentes salles de marché sont reliées entre elles par un réseau dédié, le *réseau inter-boursier* (voir fig. 1).

Supposons maintenant que le commerçant lambda souhaite émettre un ordre de transaction depuis son terminal de la bourse de Paris concernant une action cotée à la bourse de Londres. Les opérations suivantes ont lieu :

1. Le terminal se connecte sur le central de sa salle de marché (Paris) et émet l'ordre de transaction.
2. Le central de Paris regarde l'identificateur (*i.e.*, le numero unique associé au nom) de l'action concernée par l'ordre. Se rendant compte que cette action n'est pas coté à Paris, il achemine l'ordre vers le central de Londres, au travers du réseau inter-boursier.

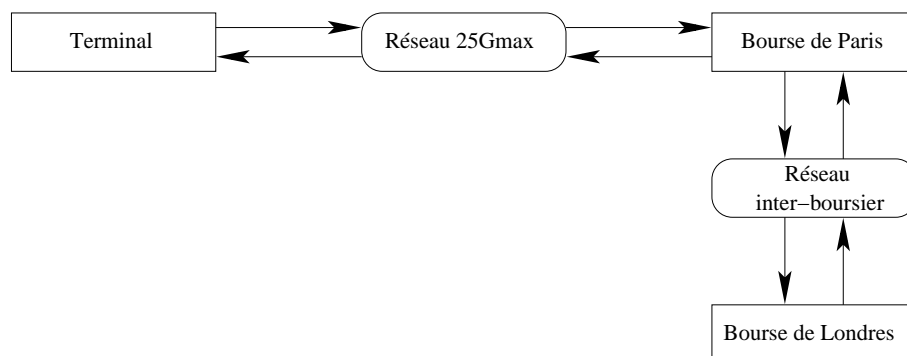


FIGURE 1 – Principe de communication des ordres et des différents réseaux

3. Le central de Londres prend connaissance de l'ordre de transaction, consulte la cotation actuelle de l'action concernée et vérifie que la transaction peut être effectuée (c'est-à-dire que le stock d'actions disponibles permet de satisfaire la demande).
4. Si la transaction peut être effectuée, il répond au central de Paris (toujours via le réseau inter-boursier) que l'opération peut être effectuée et envoie le montant global correspondant à la transaction demandée. Si ce n'est pas le cas, il répond le contraire.
5. Enfin le central de Paris transmet la réponse au terminal du commerçant.
6. L'ordre est validé pour le montant transmis dans la réponse de Londres (notez que le commerçant ne se soucie jamais de savoir s'il a suffisamment d'argent ; c'est un comportement très réaliste chez les commerçants!).

1.3 Le central des salles de marchés

La suite d'opérations décrite ci-dessus montre que le rôle du central des salles de marché est double. En effet, l'action du central diffère selon si l'ordre est local ou non. i) Si ce n'est pas le cas, son rôle est d'acheminer la demande vers le central qui a la capacité de traiter l'ordre (*cf.* étape 2 ci-dessus). ii) Si la demande de transaction est locale, son rôle est de vérifier que l'ordre peut être exécuté (*cf.* étape 4 ci-dessus) et de produire un accusé de réception qui est ensuite acheminé vers le terminal depuis lequel l'ordre a été émis.

Chaque central est donc composé de deux serveurs.

Le **serveur d'acquisition**. Il s'agit du serveur de la salle de marché auquel se connectent les terminaux. Une fois connecté, les terminaux envoient au serveur d'acquisition toutes les informations concernant les ordres de transaction.

Le **serveur d'exécution**. Il s'agit du serveur auquel le serveur d'acquisition transmet les ordres de transaction concernant les actions cotés dans cette salle de marché. C'est lui qui est chargé de produire l'accusé de réception. L'accusé

de réception suit donc le chemin inverse, c'est-à-dire : serveur d'exécution → serveur d'acquisition → terminal.

1.4 Le routage

Pour effectuer le routage des ordres de transaction, c'est-à-dire pour déterminer à quel serveur central chaque ordre doit être transmis, les serveurs d'acquisition utilisent les premiers numéros de l'identifiant de chaque action : ceux-ci indiquent la salle de marché dans laquelle l'action est cotée.

Chaque serveur d'acquisition analyse donc le numéro de l'action concernée par l'ordre, puis :

- si l'action est cotée dans la même bourse que celle du serveur d'acquisition, il envoie l'ordre au serveur d'exécution auquel il est connecté ;
- si l'action est cotée dans une autre bourse, le serveur envoie l'ordre sur le réseau inter-boursier, sans se préoccuper de la suite du transit.

Le réseau inter-boursier n'est donc pas un simple réseau physique : il doit aussi effectuer le routage des ordres de transaction, c'est-à-dire analyser les messages qui lui sont transmis, envoyer chaque demande vers le serveur d'acquisition du réseau correspondant et, enfin, prendre en charge la transmission de l'accusé de réception lorsqu'il lui revient.

2 Cahier des charges

L'objectif de ce projet est de simuler les mécanismes décrits ci-dessus :

- le terminal envoyant un ordre de transaction vers le serveur d'acquisition ;
- le serveur d'acquisition effectuant le routage des ordres de transactions vers son propre serveur d'exécution ou le réseau inter-boursier, et effectuant le routage des accusés de réception qu'il reçoit ;
- le serveur d'exécution traitant les ordres de transaction et émettant les accusés de réception ;
- le réseau inter-boursier auquel sont connectés les différents serveurs d'acquisition, capable d'effectuer le routage des ordres de transaction et des accusés de réception ;
- le tout permettant un maximum de parallélisme.

Remarque : cet exercice est avant tout "académique", mais n'est pas dénué d'intérêt ; en effet, des sociétés commercialisent toutes sortes de simulateurs pour tester le fonctionnement des nouveaux composants, afin de valider leur fonctionnement avant leur mise en production.

Le cahier des charges fonctionnelles précise l'architecture générale, les fonctionnalités devant être programmées ainsi que les contraintes fonctionnelles à respecter. Le cahier des charges techniques fournit les restrictions concernant la mise en œuvre.

2.1 Cahier des charges fonctionnelles

2.1.1 Architecture fonctionnelle

Le schéma 2 précise celui qui a été présenté plus haut et retranscrit la description que nous avons fournie ci-dessus.

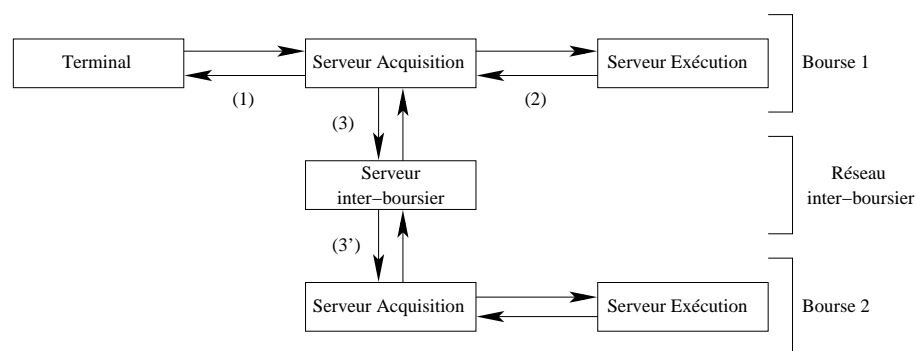


FIGURE 2 – Architecture fonctionnelle du projet

Chaque terminal est relié par 25Gmax (1) au serveur d'acquisition de la salle de marché auquel il appartient. Celui-ci est relié (2) au serveur d'exécution de cette même salle de marché.

Le réseau inter-boursier relie (3,3') les serveurs d'acquisition des différentes bourses. Tous les autres serveurs d'acquisition des différentes salle de marché d'EuroPrevious sont également reliées au réseau inter-boursier, mais ne sont pas représentées sur ce schéma.

2.1.2 Terminal

Dans le cadre de ce projet, il n'est pas question d'utiliser de vrais terminaux ni de faire appel à de vrai commerçants pour passer les ordres d'exécution. Un commerçant est autorisé à se connecter dans toutes les bourses EuroPrevious grâce à la dernière technologie Virtual-Desktop. Un terminal étant un moyen d'envoyer un ordre de transaction, il est simulé pour ce projet par un exécutable qui enverra un ordre aléatoire émis par un commerçant également tiré aléatoirement parmi la liste de tous les commerçants EuroPrevious disposant d'un compte Virtual-Desktop. Dans le système Virtual-Desktop, chaque commerçant est identifié par un code à 10 chiffres.

Pour fonctionner, un terminal a donc besoin de connaître la liste des actions disponible dans l'ensemble des bourses du groupe EuroPrevious et la liste des commerçants disposant d'un identificateur Virtual-Desktop.

Chaque terminal envoie les ordres générés vers son serveur d'acquisition.

Lorsqu'un terminal reçoit (ultérieurement) une réponse du serveur, il affiche cette réponse à l'écran (ordre accepté ou refusé et montant de la transaction).

Les échanges entre les terminaux et leur serveur ont lieu suivant un protocole bien déterminé : les informations sont formatées d'une certaine façon. Afin de simplifier le projet et de garantir l'interopérabilité des différents projets entre eux (voir en annexe pour une explication de ce point), un seul protocole de communication est utilisé. Celui-ci est décrit en annexe de ce document.

2.1.3 Serveur d'acquisition

Un serveur d'acquisition n'a qu'une fonction de routage.

- Il doit pouvoir accepter des ordres de transaction provenant des terminaux et du réseau inter-boursier.
- Il doit pouvoir effectuer le routage des ordres de transaction vers le serveur d'exécution ou vers le réseau inter-boursier.
- Il doit pouvoir accepter les accusés de réception provenant du serveur d'exécution ou du réseau inter-boursier.
- Il doit pouvoir envoyer les accusés de réception vers les terminaux ou le réseau inter-boursier.

Un serveur d'acquisition doit donc être capable d'utiliser le protocole de communication employé par les terminaux, ainsi que le protocole du réseau inter-boursier (voir les protocoles définis en annexe).

Afin d'effectuer correctement le routage des ordres de transaction, le serveur d'acquisition analyse, pour chaque ordre de transaction, l'identificateur (codé sur 10 chiffres) de l'action concernée par l'ordre : les 4 premiers chiffres indiquent le lieu de cotation de l'action.

Afin d'effectuer correctement le routage des accusés de réception, un serveur d'acquisition doit garder trace des commerçants connectés à son réseau et attendant un accusé de réception à un ordre envoyé.

2.1.4 Serveur d'exécution

Le serveur d'exécution doit être capable de fournir une réponse à un ordre de transaction. Il doit donc déterminer si l'ordre est compatible avec le stock courant et dans le cas positif calculer le montant correspondant à la transaction demandée. Dans le cadre de ce projet, le serveur d'exécution a donc accès :

- au nombre d'actions actuellement stockées dans la bourse, pour chaque type d'actions ;
- au nombre maximum d'actions que la bourse peut stocker ; et
- à la valeur intrinsèque de chaque action (qui sert à calculer les prix d'achat / vente des actions en fonction du stock).

Lorsque le serveur d'exécution reçoit un ordre d'achat d'actions, il compare son stock au nombre d'actions N demandé dans l'ordre. Dans le cas où le stock est supérieur ou égal à N , il effectue les opérations suivantes :

- calcul du prix d'achat des N actions (voir l'annexe sur la cotation) ;
- envoi de l'accusé de réception positif comprenant le montant global de la transaction ; et
- mise à jour du stock.

Dans le cas où le stock est insuffisant (inférieur à N), le serveur d'exécution envoie un accusé de réception négatif.

Lorsque le serveur d'exécution reçoit un ordre de vente d'actions, il vérifie qu'il a la capacité pour stocker ces nouvelles actions. Si c'est le cas, il effectue les opérations suivantes :

- calcul du prix de vente des actions (voir l'annexe sur la cotation) ;
- envoi de l'accusé de réception positif comprenant le montant global de la transaction ; et
- mise à jour du stock.

Dans le cas où le serveur n'a pas la capacité pour stocker les actions proposées par le commerçant, il envoie un accusé de réception négatif.

2.1.5 Le réseau inter-boursier

Le réseau inter-boursier n'a qu'une fonction de routage :

- il doit pouvoir accepter les messages (ordres de transaction et accusés de réception) provenant des serveurs d'acquisition ;
- il doit pouvoir analyser le contenu des messages pour déterminer vers quel serveur d'acquisition il doit les envoyer.

2.2 Cahier des charges techniques

2.2.1 Contraintes générales

- L'ensemble de la simulation tournera sur une seule machine.
- Les programmes seront écrits en langage C, et mettront en œuvre les concepts et les techniques apprises pendant le cours IN301/IN3ST01.
- Un maximum de parallélismes est exigé dans ce projet.

2.2.2 Mémoire des serveurs

Pour gérer les ordres en attente, les serveurs d'acquisition ont une mémoire finie. La taille de cette mémoire (le nombre de case du tableau) sera précisée sur la ligne de commande au lancement d'un serveur.

2.2.3 Nombre de processus

Chaque composant "fonctionnel" tel qu'il a été présenté dans le cahier des charges fonctionnelles correspond à un processus. On trouve donc un processus pour le terminal (que l'on nommera *Terminal*), un processus pour un serveur d'acquisition (*Acquisition*), un processus pour un serveur d'exécution (*Execution*) et un processus pour le réseau inter-boursier (*Inter-boursier*).

La simulation pouvant mettre en jeu plusieurs terminaux et plusieurs salles de marché, etc., on trouve en fait un processus *Terminal* par terminal, un processus *Acquisition* par salle de marché et un processus *Execution* par salle de marché.

Pour favoriser le parallélisme, chaque processus peut (si besoin est) être découpé en plusieurs threads.

2.2.4 Paramètres

Les paramètres nécessaires au fonctionnement des différents processus sont fournis via “la ligne de commande”, à l’exception des paramètres suivants qui sont fournis sous forme de fichier :

- Un annuaire des bourses : la liste des salles de marché et de leur code à 4 chiffres (par exemple 0001 pour Paris, 0002 pour Londres, etc.);
- Un annuaire global de toutes les actions existantes : la liste des identificateurs des actions à 10 chiffres (dont les quatre premiers encodent le lieu de cotation), leur valeur intrinsèque et le stock initial disponible en bourse.
- Un annuaire des commerçants ayant accès à EuroPrevious via la technologie Virtual-Desktop : la liste des commerçants et de leur identificateur codé sur 10 chiffres.

Ces fichiers seront au format texte, chaque ligne contenant les informations demandées (nom de la bourse et code de 4 chiffres ; identificateur de l’action, valeur intrinsèque et stock ; nom du commerçant et code à 10 chiffres), séparées par des espaces.

2.2.5 Gestion des échanges par tuyaux

Chaque terminal est connecté à son serveur d’acquisition par une paire de tuyaux. Le serveur a donc comme rôle d’orchestrer simultanément les lectures et les écritures sur ces tuyaux. Ceci implique de maintenir une table de routage entre message et terminaux pour chacun des processus *Acquisition*.

Les échanges entre un serveur d’acquisition et un serveur d’exécution sont possibles au travers d’une paire de tuyaux.

Pour ce qui concerne le réseau inter-boursier et les serveurs d’acquisition, la problématique est strictement identique à celle des terminaux : il faut utiliser une paire de tuyaux pour connecter chaque serveur d’acquisition au réseau inter-boursier. Ceci implique donc également que le processus *inter-boursier* maintient une table de routage entre serveurs d’acquisitions et messages.

2.3 Comment tester sans s’y perdre ?

Le schéma proposé ci-dessus comporte un petit défaut tant que les communications entre processus se feront sur la même machine (donc sans utiliser de *socket*, développement proposé en option) : chaque *Terminal* va fonctionner à partir de la même fenêtre `xterm` (le même terminal, le même *shell*). Tous les affichages vont donc se faire sur cette même fenêtre.

Afin de faire bénéficier chaque processus *Terminal* d’une entrée et d’une sortie standard qui lui soient propres, une astuce peut être utilisée : lors de la création d’un nouveau *Terminal* par le processus *Acquisition*, recouvrir le nouveau fils du processus non pas par *Terminal*, mais par `xterm -e Terminal`.

Cette astuce n'est pas très esthétique, et n'est qu'un intermédiaire avant la communication par socket.

2.4 Livrables

Doivent être livrés sous format électronique :

1. Un ensemble de fichier C, amplement commentés, correspondant aux différents programmes constituant la simulation.
2. Un fichier Makefile permettant de compiler tous les programmes (y compris les programmes de test).
3. Un mode d'emploi expliquant comment obtenir une application opérationnelle, comment l'exécuter, et comment la tester.
4. Un manuel technique détaillant l'architecture, le rôle de chaque composant, expliquant comment tester le bon fonctionnement de façon indépendante et justifiant les choix techniques effectués. Ce manuel doit en particulier bien préciser comment le projet répond au cahier des charges (ce qu'il sait faire, ce qu'il ne sait pas faire, et mettre en exergue ses qualités et ses défauts).
5. Dans le manuel technique, on démontrera qu'il ne peut pas y avoir d'interblocage entre les différents processus, pas plus qu'entre les threads d'un même processus.

Tous les documents à produire s'adressent à des ingénieurs généralistes et les rédacteurs doivent donc veiller à donner des explications concises et claires.

Aucun fichier exécutable, fichier objet, fichier de sauvegarde, fichier superflu ou inutile ne devra être transmis avec les livrables.

Une version papier des rapports sera également remise au secrétariat.

3 Conduite du projet

Les étapes qui vous sont suggérés dans cette partie permettent une approche "sereine" de la programmation de ce projet.

3.1 Introduction

3.1.1 Un projet en plusieurs étapes

Le développement du simulateur présenté ici doit être réalisé en plusieurs étapes. Ce découpage conduit le développeur à écrire des programmes indépendants les uns des autres, qui communiqueront entre eux par l'intermédiaire de tuyaux, souvent après redirection des entrées/sorties standards.

3.1.2 Méthode “diviser pour régner”

La constitution de l’application globale par écriture de “petits” programmes indépendants qui communiqueront ensuite entre eux est un gage de réussite : chaque “petit” programme générique peut être écrit, testé et validé indépendamment, et la réalisation du projet devient alors simple et progressive.

La meilleure stratégie à adopter consiste à écrire une version minimale de chaque brique du projet, à faire communiquer ces briques entre elles, puis éventuellement, à enrichir au fur et à mesure chacune des briques. La stratégie consistant à développer à outrance une des briques pour ensuite s’attaquer tardivement au reste du projet conduit généralement au pire des rapport résultat/travail.

Avant toute programmation, conception ou réalisation de ce projet, il est fortement conseillé de lire l’intégralité de l’énoncé, y compris les parties que vous pensez ne pas réaliser, et de s’astreindre à comprendre la structuration en étapes proposée ici.

La traduction de cet énoncé sous forme de schéma est indispensable.

En particulier, il est particulièrement important de définir dès le départ l’ensemble des flux de données qui vont transiter d’un processus à un autre, de déterminer comment il est possible de s’assurer que ces flux sont bien envoyés et reçus, puis de programmer les fonctions d’émission et de réception correspondantes. Ces fonctions seront ensuite utilisées dans tous les processus du projet.

Un schéma clair et complet associé à des communications correctement gérées garantissent la réussite de ce projet et simplifient grandement son développement.

3.2 Étape 1 : les fonctions de communication

La première étape a été faite pour vous. Elle consiste à formater (selon le protocole définis en annexe) les ordres d’exécution et les accusés de réception un message complet à partir de différents champs et réciproquement à extraire les différents champs à partir d’un message déjà formaté. Les messages ainsi formatés peuvent être lus et écrits dans des fichiers grâce aux librairies programmées pendant les TD machines et les TP.

Les problèmes de synchronisation seront abordés dans les étapes suivantes et il s’agit pour l’instant uniquement de traduire sous forme de programmes (de fonctions en C) les protocoles de communication : formattage de messages selon la structure définie en annexe et récupération des informations stockées sous cette forme.

Les fonctions qui ont été faites pour vous sont téléchargeables en suivant le lien donné en annexe. Une fois que vous aurez lu le code, et compris son utilisation, vous serez à même de les utiliser ; ces fonctions sont utilisées par tous les processus du projet.

3.3 Étape 2 : réalisation de programmes indépendants

Dans cette seconde étape, il s'agit de mettre en place les différentes briques du projet et de pouvoir les tester indépendamment.

3.3.1 Processus : *Terminal*

Le processus *Terminal* offre l'interface permettant d'envoyer des ordres et de recevoir les accusés de réception. Nous souhaitons utiliser le même code pour tester *Terminal* de manière indépendante et pour l'utiliser dans la simulation globale du système. Pour cela, *Terminal* accepte comme arguments deux descripteurs de fichier vers lesquels il redirige son entrée et sa sortie standard (voir annexe). Ensuite, il lit son entrée standard et écrit sur sa sortie standard. On peut ainsi utiliser *Terminal* :

- en passant 0 et 1 comme arguments, ce qui permet de tester *execution* “à la main” en lisant les ordres de transaction via la sortie standard et en lui passant les accusés de réception sur l'entrée standard ; ou
- en passant les descripteurs de tuyaux utilisés par *Acquisition*, ce qui permet d'utiliser *Terminal* dans la simulation complète du système.

3.3.2 Processus : *Execution*

Le processus *Execution* offre l'interface permettant de recevoir des ordres et d'envoyer les accusés de réception. Comme *Terminal*, il accepte comme arguments deux descripteurs de fichier vers lesquels il redirige son entrée et sa sortie standard.

La méthode pour calculer les prix d'achat et de vente des actions est décrite en annexe. Les programmes permettant de calculer ces prix vous sont fournis en suivant le lien de cette annexe.

3.3.3 Processus : *Acquisition*

Le processus *Acquisition* reçoit des ordres de transaction en provenance soit des terminaux, soit du réseau inter-boursier et des accusés de réception en provenance soit du réseau inter-boursier soit du serveur d'exécution. Les ordres seront redirigés (“routés”) vers le réseau ou bien vers le serveur d'exécution et les accusés seront redirigés soit vers les terminaux soit vers le réseau, conformément au cahier des charges. A cette phase du projet, on peut utiliser :

- des fichiers dans lesquels le processus *Acquisition* écrit lorsqu'il est supposé envoyer un message ;
- des fichiers dans lesquels le processus *Acquisition* lit les messages lorsqu'il s'attend à recevoir ; ces derniers seront préparés “à la main”.

Ces fichiers permettent de simuler les échanges avec les processus *Execution*, *Terminal* et *inter-boursier*.

Le programme doit accepter sur sa ligne de commande au moins un paramètre représentant la taille de la mémoire servant à la gestion des ordres de transactions.

Dans une première phase, le processus *Acquisition* traite *séquentiellement* chaque terminal, puis le serveur d'exécution et enfin le réseau inter-boursier. Dans une deuxième phase, on *parallélise* au maximum les requêtes. La première phase doit être pensée et conçue sur le papier afin de pouvoir facilement passer à la deuxième phase. La première phase peut suffire pour passer à la suite du projet, mais elle est obligatoire et devra être rendue. On conservera toutes les versions des différentes phases du programme à des fins de test.

3.4 Étape 3 : création d'un réseau Bourse

3.4.1 Préparation de la communication par tuyaux

En pratique, le processus *Acquisition*, le processus *Execution* et chaque processus *Terminal* communiquent par une paire unique de tuyaux. Une fois ces deux tuyaux créés, il est possible de modifier le programme *Acquisition* pour qu'il utilise non pas des fichiers mais ces deux tuyaux. Les processus *Terminal* et *Execution* n'ont pas à être modifiés.

3.4.2 Raccordement

Pour terminer la mise en place des communications au sein d'une même bourse, il faut maintenant créer d'une part une paire de tuyaux entre *Acquisition* et chaque *Terminal* (il y aura autant de paires de tuyaux que de terminaux) et, d'autre part, une paire de tuyaux entre *Acquisition* et *Execution*.

Modifier le programme *Acquisition* pour qu'il accepte sur sa ligne de commande les paramètres suivants :

- la taille de la mémoire de gestion des messages.
- le nom du fichier *annuaire des bourses* (ce fichier n'est pas indispensable à proprement parler, mais il sert à afficher un nom plutôt qu'un code, ce qui est plus joli) ;
- le code de 4 chiffres associés à cette bourse ;
- le nom du fichier *annuaire global des commerçants* ;
- le nom du fichier *annuaire global des actions*.

Créer les tuyaux nécessaires, opérer les clonages et recouvrements nécessaires pour créer les processus *Terminal* et *Execution* en nombre suffisant.

3.5 Étape 4 : création du réseau inter-boursier

3.5.1 Processus : *Inter-boursier*

Chaque serveur d'acquisition est relié au processus *Inter-boursier* par une paire de tuyaux. Celle-ci permet à *Inter-boursier* de recevoir les ordres et de transmettre les accusés en retour, après les avoir routés.

L'architecture à mettre en place entre *Inter-boursier* et les différents processus *Acquisition* est similaire à celle mise en place entre chaque processus *Acquisition* et les processus *Terminal* qui y sont reliés.

Dans un premier temps, les messages transitant par *Inter-boursier* sont simplement lus et écrits dans des fichiers, sans qu'aucune communication ne soit mise en place.

Dans une première phase, le processus *Inter boursier* traite *séquentiellement* chaque serveur d'acquisition. Dans une deuxième phase, on *parallélise* au maximum les requêtes. La première phase doit être pensée et conçue sur le papier afin de pouvoir facilement passer à la deuxième phase. La première phase peut suffire pour passer à la suite du projet, mais elle est obligatoire et devra être rendue. On conservera toutes les versions des différentes phases du programme à des fins de test.

3.5.2 Raccordement

On crée un programme *Gestion* qui doit accepter sur sa ligne de commande un fichier de configuration dans lequel on trouve les informations suivantes :

- la taille de la mémoire de gestion des message d'un central ;
- le nom du fichier *annuaire des bourses* ;
- le nom du fichier *annuaire global des commerçants* ;
- le nom du fichier *annuaire global des actions*.

Le processus *Gestion* devra créer les tuyaux, effectuer les clônages et les recouvrements nécessaires afin de créer l'ensemble des processus nécessaires à la simulation.

4 Évolutions complémentaires et optionnelles

4.1 Ouverture et clôture de séances

En pratique les bourses ne sont pas ouvertes 24 heures sur 24 : les cotations sont interrompues chaque soir. Quel mécanisme peut être mis en œuvre pour signaler le début et la fin de la journée ? Proposez et implémentez une solution tenant compte des horaires d'ouverture des salles de marché.

Cette évolution du projet incite également à effectuer chaque soir le bilan comptable de chaque commerçant. De quelles informations a-t-on besoin chaque soir pour effectuer ce bilan ? Proposez et mettez en œuvre un mécanisme qui intègre cette fonctionnalité à votre simulation.

4.2 Bilan des entreprises cotées en bourse

Jusqu'ici, la valeur intrinsèque des entreprises ne varie pas. Dans la réalité les entreprises effectuent un bilan chaque trimestre. Cela permet de savoir si elles valent plus au moins que le trimestre précédent. Proposer et mettez en œuvre un mécanisme signalant la fin de chaque trimestre et permettant de changer la valeur intrinsèque des entreprises à ce moment là.

4.3 Utilisation de socket

L'utilisation de communications entre machines ne fait pas partie des objectifs de ce cours. Cependant, afin de rendre le projet plus vivant et plus attractif, nous proposons aux plus débrouillards d'utiliser des `socket` de sorte que la simulation soit plus réaliste :

- chaque terminal communique avec son serveur d'acquisition par des `socket` ;
- chaque serveur d'acquisition communique avec le réseau inter-boursier par `socket`.

Les informations nécessaires à l'utilisation des `socket` peuvent facilement se trouver sur internet.

Le travail de mise en œuvre des communications par `socket` doit être entrepris **uniquement après avoir réussi à programmer une application complètement opérationnelle en mode texte.**

Attention : **ne jamais modifier un programme qui fonctionne et toujours travailler sur une copie de celui-ci.**

5 Annexes

5.1 Codes disponibles en ligne

L'archive, que vous trouverez en ligne à l'adresse ci-dessous, fournit certaines fonctions C (décrites Sections 5.3, 5.4 et 5.5) qui pourront vous aider dans la réalisation du projet.

`http://www.esiee.fr/%7Einfo/a2si/TC-ESIEE/IN301ST01/libbourse.tgz`

Vous trouverez également un générateur aléatoire implémenté dans les fichiers `alea.c` et `alea.h`.

Pour décompresser l'archive, tapez la commande `'tar zxvf libbourse.tgz'` après l'avoir enregistrée dans votre répertoire courant.

5.2 De l'intérêt des standards pour assurer une meilleure interopérabilité

Le recours à des protocoles "standardisés" (ou pour le moins communs) à un double intérêt :

- il permet de gagner du temps sur le développement de ce projet et d'illustrer par la pratique la façon dont les problèmes sont traditionnellement résolus en informatique ;
- il permet de mélanger les projets entre eux afin de tester le respect du protocole et peut-être, d'aider certains binômes à avancer (il suffit d'utiliser les processus d'un autre binôme²).

2. Bien entendu, cette utilisation ne peut s'entendre qu'à des fins de mise au point...

La capacité ainsi esquissée à mélanger des composants issus de programmeurs ou prestataires différents pour constituer un système d'information unique se nomme "interopérabilité". C'est un des enjeux majeurs de l'informatique actuelle.

5.3 Protocoles de communication et format de messages

Les messages sont constitués de caractères ASCII (sans accent). Chaque message est constitué de différents champs séparés par les caractères '|' et terminés par un caractère de fin de message '\n'.

Remarque : afin de simplifier au maximum le protocole, on considère qu'il est impossible que plus d'un ordre de transaction, effectué par un commerçant donné, ne circule sur le réseau. Cette hypothèse, assez réaliste, permet de simplifier l'appariement des ordres et des accusés de réception au niveau des processus *Acquisition* et *Inter-boursier*.

Les échanges utilisent deux types de message, *l'émission d'ordre*, et *l'accusé de réception*. Ces deux types de messages partagent un format générique :

Format : |E...E|T...T|A...A|V...V|\n

- E...E est l'identifiant de l'émetteur de l'ordre ;
- T...T est le type du message, c'est à dire 'Achat', 'Vente', 'AccuseVente' ou 'AccuseAchat' ;
- A...A est l'identificateur de l'action sur laquelle porte l'ordre ou l'accusé de réception ;
- V...V est la valeur associé au message. Dans le cas d'un ordre, il s'agit du nombre d'actions concernées par l'ordre. Dans le cas d'un accusé de réception, il s'agit de la valeur pour laquelle l'ordre a été effectué si l'accusé de réception est positif ou de '0' si l'accusé de réception est négatif.

Les fichiers message.c et message.h (fournis dans l'archive disponible en ligne) implémentent les fonctions pour créer et interpréter un message formaté selon ce protocole.

5.4 Cotation des actions en bourse

Nous décrivons ici le système de cotation des actions en bourse. Nous avons vu que le cotation d'une action A_i dépend du stock courant S_i dans la bourse responsable de sa cotation et de sa valeur intrinsèque V_i . Pour modéliser la loi de l'offre et de la demande, on définit la cotation de cette action par V_i/S_i .

Lorsqu'un commerçant achète une action il paye V_i/S_i et le stock devient $S'_i = S_i - 1$. Donc, si le commerçant achète immédiatement après une seconde action il paiera un prix différent : $V_i/S'_i = V_i/(S_i - 1)$. Il faut tenir compte de ce phénomène lorsqu'un commerçant effectue un ordre d'achat de N actions (avec $N > 1$). Le **prix d'achat** PA_i^N **de** N **actions** A_i est donc :

$$PA_i^N = \sum_{k=0}^{N-1} V_i/(S_i - k).$$

Lorsqu'un commerçant achète une action et qu'il la revend immédiatement après, il semble raisonnable qu'il n'en tire ni perte ni bénéfice. Pour cela, il est

nécessaire que le prix de vente d'une action diffère de son prix d'achat. Plus précisément, le **prix de vente** PV_i^N de N actions A_i est :

$$PV_i^N = \sum_{k=1}^{k=N} V_i / (S_i + k).$$

Les fichiers PrixAchatVente.c et PrixAchatVente.h (fournis dans l'archive disponible en ligne) implémentent les fonctions pour calculer les prix d'achat et de vente d'actions selon le système EuroPrevious.

5.5 Redirection des entrées et des sorties

La redirection des entrées et des sorties peut se faire grâce à l'appel système `dup`.

```
int dup2(int oldfiledes , int newfiledes);
```

Cet appel système sert à dupliquer le contenu du descripteur `oldfiledes` dans un autre descripteur `newfiledes`. Il renvoie -1 en cas d'échec, et il prend en argument deux descripteurs de fichiers. Par exemple si on a envie que STDIN (==1) soit en fait STDERR (==2) on peut écrire ceci :

```
dup2(2 , 1);
```

L'exemple suivant redirige l'entrée et la sortie standard vers un tube, et recouvre le processus courant et son fils par deux programmes. Le père et le fils vont communiquer par leur entrée et sortie standard.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

void usage(char * basename) {
    fprintf(stderr ,
        "usage : %s [<programme_1>] [<programme_2>]\n" ,
        basename);
    exit(1);
}

int main(int argc , char *argv[]) {
    int pid; /* permet d'identifier qui on est*/
    int fdpipe[2]; /* sera utilisé pour lier les processus */

    if (argc != 3) usage(argv[0]);

    /* on créé le pipe qui sera utilisé pour relier
       la sortie du premier processus
       vers l'entrée du second
    */
    if ( pipe(fdpipe) == -1 ) {
        perror("pipe");
        exit(-1);
    }
}
```

```

}

switch(pid = fork()) {
  case -1:
    /* le fork a échoué */
    perror("fork");
    exit(-1);
  case 0:
    /* code du fils */
    /* on fait en sorte que lorsque le processus
       écrira sur la sortie standard (1)
       il le fera en fait dans le pipe (fdpipe[1])
    */
    dup2(fdpipe[1], 1);
    /* on ferme tout, même le pipe...
       on n'en a plus besoin
    */
    close(fdpipe[0]);
    close(fdpipe[1]);
    execlp(argv[1], argv[1], NULL);
    /* pas besoin de break,
       ce code n'existe déjà plus à l'exécution
    */
  default :
    /* code du père */
    /* on fait en sorte que lorsque le processus
       lira sur l'entrée standard (0)
       il le fera en fait dans le pipe (fdpipe[0])
    */
    dup2(fdpipe[0], 0);
    close(fdpipe[0]);
    close(fdpipe[1]);
    execlp(argv[2], argv[2], NULL);
}
/*
cette portion de code ne sera jamais exécutée,
puisque les processus ont déjà
été remplacé. On met néanmoins un
return sinon le compilateur proteste.
*/
return 0;
}

```

Le code ci-dessus permettant de rediriger les entrées et sorties est disponible dans le fichier TestRedirection.c que vous trouverez dans l'archive téléchargées.