# L3 - Interopérabilité – TP1, Client/serveur

## Interopérabilité « manuelle » via un mécanisme client/serveur

Dans ce TP, nous allons appeler une fonction C/C++ depuis un programme Java sans utiliser aucun mécanisme d'interopérabilité fourni par le langage Java.

Ce TP va nous permettre de faire deux choses.

**La première**, faire communiquer Java et C++ sans mécanisme offert par la Java Virtual Machine. Nous constaterons la lourdeur de l'implémentation client/serveur, avec beaucoup de code dit « boilerplate » (le code général qui doit être écrit mais qui ne représente pas de réelle valeur ajoutée). Ce code est ici le code dit « proxy/stub » permettant à 2 composants hétérogènes de communiquer. Nous verrons ces notions en cours.

Lors des prochains TP, nous utiliserons des mécanismes dédiés pour communiquer sans cette lourdeur.

**La seconde**, la lourdeur observée ici va nous permettre d'introduire le prochain cours également sous un autre angle. Ce cours expliquera que, même lorsque l'on doit communiquer entre 2 processus distincts, parfois même depuis 2 machines différentes, il reste possible de le faire de manière simple, en supprimant la lourdeur client/serveur, en utilisant certains outils.

## Mécanisme client/server utilisé

Il y a plusieurs moyens simples de faire communiquer 2 processus, parmi lesquels :

- un fichier de communication avec un système de surveillance de modifications (l'OS fourni cette fonctionnalité qui permet d'être averti quand un fichier est modifié)
- un socket TCP
- des appels RPC
- de nombreuses autres méthodes

Nous choisissons la première option ici. La méthode utilisée importe peu.

## Premier lancement

Suivez les étapes suivantes :

- depuis l'espace community, téléchargez le code de démarrage pour le TP1. Il y a 2 projets, le projet client en Java (TP1ClientJavaProject), et le projet serveur en C++ (TP1ServerCPPProject).
- Décompactez ces projets dans votre Workspace Visual Studio Code.
- Depuis un terminal ou un explorateur, créez un fichier 'test.txt'.
  Sous Linux, créez-le impérativement dans un nouveau répertoire, '/tmp/TP1/'.
  Sous Windows, mettez-le où vous souhaitez, mais notez son chemin complet. Si vous le mettez dans un nouveau répertoire 'TP1' dans votre répertoire Documents, le chemin sera typiquement : C:\\Users\\<alias>\\Documents\\TP1\\test.txt
- Ouvrez ce fichier 'test.txt' avec l'éditeur de Visual Studio Code (attention, ce fichier est situé hors de votre workspace), tapez un peu de contenu dedans, sauvez, mais laissez l'éditeur ouvert.
- Editez le fichier Main.java et spécifiez le chemin d'accès au fichier test.txt que vous avez noté précédemment.

A présent :

- Lancez le programme Java et observez la sortie texte de ce programme. Vous devez voir une chaine « do_something » qui apparaît. Elle a été écrite dans le fichier test.txt.
- A présent tapez par exemple « bonjour » dans le fichier test.txt, depuis l'éditeur, et faites attention au moment où vous sauvez (Ctrl+S) : le programme Java réagit, réceptionne la chaîne de caractères tapée, et poursuit son exécution.
- Un va et vient (write/read) est à nouveau engagé (modifiez à nouveau le fichier test.txt dans l'éditeur)
- Le programme se termine.

## Prise en main du programme C++

Examinez le code source du programme C++. Un seul fichier source vous intéresse, il s'agit de TP1ServerCPPProject/src/efsw/**main.cpp**. Vous pouvez ignorer tous les autres fichiers.

Editez le fichier main.cpp et spécifiez le chemin d'accès au fichier test.txt que vous aviez noté plus haut. Attention, le code C++ surveille le répertoire parent du fichier test.txt, donc le répertoire 'TP1' que vous avez créé ; pas le fichier 'test.txt' lui-même.

Faites le build du projet C++ en vous aidant de la procédure décrite dans les fichiers sur community. Il y a 2 fichiers de documentation relativement au C++, que vous devez suivre et maîtriser. Ils serviront pour d'autres TP ainsi que pour le projet.

Lancez le programme C++. Le programme C++ reprend les mêmes principes que le programme Java. Vous pouvez mettre des points d'arrêt pour étudier plus en avant son exécution.

## Exercice 1

Vous allez maintenant utiliser ce système pour appeler une fonction C++, définie par vos soins, depuis Java.

- en Java, codez une fonction « direBonjour », qui prend en entrée un *prénom*, et qui renvoie une chaîne de caractères « Bonjour, <*prenom*> ! » (le programme C++ ne fait pas le print ; il construit la chaîne pour dire bonjour, et c'est Java en retour, qui fera le print) :

  ```
  String direBonjour(String prenom) {
    …
  }
  ```

- Dans le code C++, définissez la même fonction

  ```
  std::string direBonjour(std::string prenom) {
    …
  }
  ```

  implémentez-là (facile !)

- A présent écrivez le code Java permettant de communiquer avec le serveur C++, et écrivez le code C++ permettant de lire les données d'entrée (le prénom) depuis le fichier, et d'écrire le résultat de la fonction dans ce dernier.

## Exercice 2

Ajoutez un argument après le prénom, représentant l'âge, défini sous forme d'entier. Changez les implémentations Java et C++ en conséquence.

## Exercice 3

Modifiez à nouveau la fonction direBonjour pour qu'elle prenne en unique paramètre :

```
class Personne {
  String nom ;
  String prenom ;
  int age ;
}
```

et en C++ :

```
typedef struct {
  String nom ;
  String prenom ;
  int age ;
} Personne ;
```

Modifiez le code des deux projets pour que la mécanique pour l'appel avec passation de paramètre et récupération de valeur de retour soit là encore convenablement mise en place. Testez.