TP 1 : Connexion à distance + Fonctionnalités MPI de bases

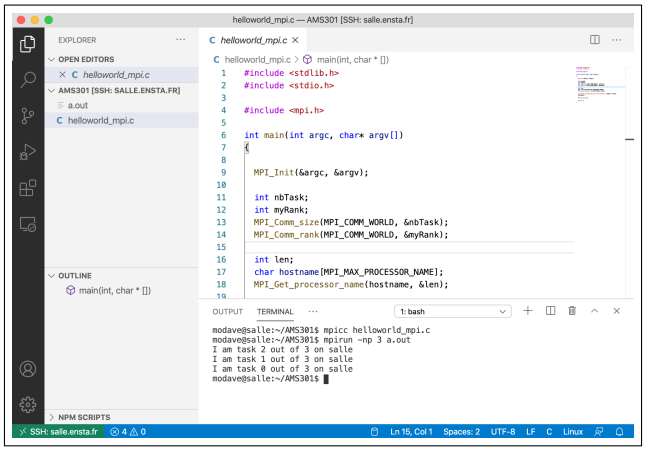
L'objectif de cette séance est double :

1. Installer les outils qui seront nécessaires pour le cours, et se familiariser avec un environnement pour travailler à distance sur une machine de calcul de l'école;
2. Apprendre à utiliser les fonctionnalités de base de MPI.

Les exercices seront réalisés à distance sur les stations de travail de l'école. Les codes sont disponibles sur le site Internet du cours.

Partie 1 – Installation des outils et connexion à distance

Pour travailler à distance sur une station de travail de l'école, il est recommandé d'utiliser le programme VS Code avec l'extension "Remote SSH". Vous pourrez parcourir/éditer vos codes à distance, ainsi que les compiler/exécuter à distance grâce à un terminal intégrée dans l'interface.



1. Installer VS Code sur votre machine personnelle :
   https://code.visualstudio.com/
2. Installer l'extension "Remote SSH" :
   https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh
3. Après installation, cliquer sur l'icône  du menu de droite de VS Code pour accéder à l'interface de connexion à distance. Dans la colonne de droite, vous verrez apparaître les différentes connexions *ssh* qui sont déjà configurées sur votre machine.
4. Dans l'interface de connexion, ajouter une nouvelle connexion *ssh* en cliquant sur le symbole +, qui apparait lorsque le curseur se trouve sur la colonne de gauche. Voici la ligne à donner :

   ```
   ssh -J IdENSTA@relais.ensta.fr IdENSTA@MachineENSTA
   ```

   Dans cette commande, `IdENSTA` doit être remplacé par votre identifiant, et `MachineENSTA` doit être remplacé par le nom de la machine qui vous sera attribué pour le cours.
5. Tester la connexion.

<u>Bonus.</u> Quelques commandes utiles lorsque vous êtes connectés sur machine

- Télécharger un fichier depuis Internet : `wget https://address-of-file`
- Décompresser une archive : `tar -xf archive.tar`
- Vérifier qui est connecté à la machine : `who`
- Vérifier les programmes qui tournent sur la machine : `top`

Partie 2 – Evaluation de la performance

Exercice 1. Première utilisation de MPI

On fournit deux versions du programme helloworld (`helloworld_mpi.c` et `helloworld_mpi2.c` en `C`, avec l'extension `.cpp` en `C++`). Ouvrez ces fichiers et vérifiez votre compréhension de chacune des lignes. Ensuite, compilez et exécutez les programmes avec les commandes qui suivent. Comprenez-vous le résultat pour chacun des cas ?

Version `C`

```
mpicc helloworld_mpi.c; ./a.out
mpicc helloworld_mpi.c; mpirun -np 2 ./a.out
mpicc helloworld_mpi.c; mpirun -np 4 ./a.out
mpicc helloworld_mpi.c; mpirun ./a.out
gcc helloworld_mpi.c; ./a.out
```

Version `C++`

```
mpicxx helloworld_mpi.cpp; ./a.out
mpicxx helloworld_mpi.cpp; mpirun -np 2 ./a.out
mpicxx helloworld_mpi.cpp; mpirun -np 4 ./a.out
mpicxx helloworld_mpi.cpp; mpirun ./a.out
g++ helloworld_mpi.cpp; ./a.out
```

Exercice 2. Communications point-à-point bloquantes

On fournit un programme (`exchange_mpi.c` en `C`, avec l'extension `.cpp` en `C++`) où deux tableaux sont échangés par les deux premiers processus MPI en utilisant les fonctions `MPI_Send` et `MPI_Recv`. Ainsi, chacun de ces processus envoie un tableau et en reçoit un autre.

On vous demande de tester ce programme en modifiant l'ordre d'appel des fonctions d'envoi et de réception, pour des tableaux de petite taille (10) et de grande taille (10000).

	Processus 0	Processus 1
Cas 1	`MPI_Send`, puis `MPI_Recv`	`MPI_Recv`, puis `MPI_Send`
Cas 2	`MPI_Recv`, puis `MPI_Send`	`MPI_Send`, puis `MPI_Recv`
Cas 3	`MPI_Recv`, puis `MPI_Send`	`MPI_Recv`, puis `MPI_Send`
Cas 4	`MPI_Send`, puis `MPI_Recv`	`MPI_Send`, puis `MPI_Recv`

Comprenez-vous le résultat pour chacun des cas ?

Exercice 3. Schema de communication en anneau

Implémentez un programme où une valeur entière (nommée *jeton*) est envoyée d'un processus MPI à un autre, de manière cyclique, en utilisant les fonctions `MPI_Send` et `MPI_Recv`. À chaque passage du jeton par un processus MPI, sa valeur est incrémentée, et le processus affiche son rang et la valeur du jeton.

Exemple pour un cas avec 3 processus MPI :

- le processus de rang 0 initialise le jeton à '0' et l'envoie au processus de rang 1 ;
- le processus de rang 1 incrémente le jeton à '1' et l'envoie au processus de rang 2 ;
- le processus de rang 2 incrémente le jeton à '2' et l'envoie au processus de rang 0 ;
- le processus de rang 0 incrémente le jeton à '3' et l'envoie au processus de rang 1 ;
- etc.