UNIVERSITÉ D'ORLÉANS **M1 Info**
Informatique Année **2017 - 2018**

Série de Travaux Dirigés : **1** *- MPI - MPI_Send/MPI_Recv*

Pour l'ensemble de ce TD, vous utiliserez pour les communications les deux routines ci-dessous

```
int MPI_Send(const void *buf, int count, MPI_Datatype datatype,
int dest, int tag, MPI_Comm comm)
int MPI_Recv(void *buf, int count, MPI_Datatype datatype,
int source, int tag, MPI_Comm comm, MPI_Status *status)
```

Pour faciliter les exercices, vous disposez sous Celene d'une archive *TD1.tgz* contenant la trame de tous les exercices. Vous trouverez un répertoire par exercice avec le programme principal et la signature de la fonction à écrire.

Exercice 1. hello word (répertoire Hello)

Compilez à partir du Makefile fourni et exécutez le programme hello.cpp (`make run`).

Exercice 2. Shift circulaire (répertoire Shift)

Ce répertoire contient le programme shifttableau qui prend en argument la taille locale d'un tableau distribué sur tous les processeurs. Il teste la fonction *shift_circulaire_tab*.

1. Testez ce programme avec 10 comme argument. Vérifiez qu'il s'agit d'un shift circulaire à droite du tableau.
2. Augmentez la taille et vérifiez que lorsque cette taille est trop grande la fonction ne marche plus. Pourquoi ?
3. Corrigez *shift_circulaire_tab* pour que ça fonctionne quelque soit la taille du tableau.

Exercice 3. Calcul du max et son indice (répertoire CalculMax)

Soit un tableau distribué sur *nprocs* processeurs, écrivez la fonction

```
void calcul_max(int n, int* tab, int root, int* maxtab)
```

qui permet de calculer le plus grand élément du tableau *tab* (le tableau est découpé en $nprocs$ blocs de taille n) sachant que le résultat doit être disponible au final sur le processeur *root* donné en argument du programme principal. La fonction renvoie *maxtab* qui n'est significatif que sur le processeur *root* telque maxtab[0] est le maximum demandé et maxtab[1] l'indice de ce maximum dans le tableau global. Le programme principal est lancé avec 2 arguments : la taille locale du tableau et le processeur root.

Exercice 4. Addition sur un anneau (répertoire Ring)

Soit *a* une variable de type int définie sur chaque processeur. Ecrivez la fonction qui permet de calculer la somme cumulée de toutes les variables a. Le résultat final doit être disponible uniquement sur le processeur *root* donné en argument du programme principal et chaque processeur n'a le droit qu'au plus à un envoi et au plus à une réception. La signature de la fonction est la suivante et le résultat qu'elle renvoie n'est significatif que sur le processeur *root*.

```
int addition_ring(int a, int root)
```

Exercice 5. Fonction d'échanges (répertoire DroiteGauche)

Écrivez la fonction de signature :

```
void shift_droite_gauche(int n, int* in, int* out)
```

qui suppose un tableau *in* distribué sur *nprocs* telque n ($n\%2 == 0$) soit sa taille en local. L'effet de cette fonction est de transmettre, pour chaque processeur, la première moitié de son tableau vers le processeur voisin de gauche et l'autre moitié vers le voisin de droite. Le processeur 0 n'a pas de voisin de gauche (donc rien à faire vers la gauche) et le dernier processeur n'a pas de voisin de droite. Les deux moitiés sont stockées dans le même tableau *out*, la moitié venant du voisin de droite étant après la moitié venant du voisin de gauche.

Refaire la même question que ci-dessus pour écrire la fonction

```
void shift_droite_gauche_circulaire(int n, int* in, int* out)
```

qui traite les voisins de manière cyclique : le dernier processeur est le voisin de gauche de 0, etc.