RÉPUBLIQUE FRANÇAISE
*Liberté*
*Égalité*
*Fraternité*

DOUANES & DROITS INDIRECTS

***AP3X23V1***

**CONCOURS EXTERNE ET INTERNE**

**POUR LE RECRUTEMENT D'INSPECTEURS DES DOUANES**

**DANS LA SPÉCIALITÉ TRAITEMENT AUTOMATISE DE L'INFORMATION – PROGRAMMEUR DE SYSTÈME D'EXPLOITATION**

**SESSION 2023**

# ÉPREUVE ÉCRITE D'ADMISSION N°3 (OBLIGATOIRE)

(DURÉE : 2 HEURES – COEFFICIENT 1)

## TRADUCTION ET RÉPONSE A DES QUESTIONS PORTANT SUR UN TEXTE EN ANGLAIS ISSU D'UNE REVUE OU D'UNE DOCUMENTATION INFORMATIQUE

***N.B : Seules les copies des candidats admissibles seront corrigées.***

### *AVERTISSEMENTS IMPORTANTS*

Veillez à bien indiquer sur votre copie le nombre d'intercalaires utilisés (la copie double n'est pas décomptée).

L'usage de tout matériel autre que le matériel usuel d'écriture et de tout document autre que le support fourni est **interdit**.

L'usage de tout dictionnaire est interdit.

La copie ne saurait comporter de **nom, initiales, paraphe, signature, lieu géographique ou tout autre élément ou signe distinctif** susceptibles de permettre l'identification du candidat. Le non-respect de cette consigne entraînera l'exclusion du concours.

**Toute fraude ou tentative de fraude** constatée par la commission de surveillance entraînera **l'exclusion du concours.**

Il vous est interdit de quitter définitivement la salle d'examen **avant le terme d'épreuve**.

Le présent document comporte **4 pages** numérotées.

**Après avoir procédé à la traduction en français des deux extraits surlignés ci-dessous :**

**1er extrait:** de « Since most libraries... » à « … popular libraries. »

**2e extrait :** de « The foundation of communication...» à « … point-to-point communications. »

**vous répondrez ensuite <u>en anglais</u> aux deux questions suivantes :**
**(il est demandé de rédiger au moins quinze lignes pour chaque réponse)**

**Question 1 :** How does the message passing model work ?

**Question 2 :** What is the history of the message passing model ?

---

Parallel computing is now as much a part of everyone's life as personal computers, smart phones, and other technologies are. You obviously understand this, because you have embarked upon the MPI Tutorial website. Whether you are taking a class about parallel programming, learning for work, or simply learning it because it's fun, you have chosen to learn a skill that will remain incredibly valuable for years to come. In my opinion, you have also taken the right path to expanding your knowledge about parallel programming - by learning the Message Passing Interface (MPI). Although MPI is lower level than most parallel programming libraries (for example, Hadoop), it is a great foundation on which to build your knowledge of parallel programming.

Before I dive into MPI, I want to explain why I made this resource. When I was in graduate school, I worked extensively with MPI. I was fortunate enough to work with important figures in the MPI community during my internships at Argonne National Laboratory and to use MPI on large supercomputing resources to do crazy things in my doctoral research. However, even with access to all of these resources and knowledgeable people, I still found that learning MPI was a difficult process.

Learning MPI was difficult for me because of three main reasons. First of all, the online resources for learning MPI were mostly outdated or not that thorough. Second, it was hard to find any resources that detailed how I could easily build or access my own cluster. And finally, the cheapest MPI book at the time of my graduate studies was a whopping sixty dollars - a hefty price for a graduate student to pay. Given how important parallel programming is in our day and time, I feel it is equally important for people to have access to better information about one of the fundamental interfaces for writing parallel applications (…).

Before the 1990's, programmers weren't as lucky as us. Writing parallel applications for different computing architectures was a difficult and tedious task. At that time, many libraries could facilitate building parallel applications, but there was not a standard accepted way of doing it.

During this time, most parallel applications were in the science and research domains. The model most commonly adopted by the libraries was the message passing model. What is the message passing model? All it means is that an application passes messages among processes in order to perform a task. This model works out quite well in practice for parallel applications. For example, a manager process might assign work to worker processes by passing them a message that describes the work. Another example is a parallel merge sorting application that sorts data locally on processes and passes results to neighboring processes to merge sorted lists. Almost any parallel application can be expressed with the message passing model.

Since most libraries at this time used the same message passing model with only minor feature differences among them, the authors of the libraries and others came together at the Supercomputing 1992 conference to define a standard interface for performing message passing - the Message Passing Interface. This standard interface would allow programmers to write parallel applications that were portable to

all major parallel architectures. It would also allow them to use the features and models they were already used to using in the current popular libraries.

By 1994, a complete interface and standard was defined (MPI-1). Keep in mind that MPI is only a definition for an interface. It was then up to developers to create implementations of the interface for their respective architectures. Luckily, it only took another year for complete implementations of MPI to become available. After its first implementations were created, MPI was widely adopted and still continues to be the de-facto method of writing message-passing applications.

Before starting the tutorial, I will cover a couple of the classic concepts behind MPI's design of the message passing model of parallel programming. The first concept is the notion of a communicator. A communicator defines a group of processes that have the ability to communicate with one another. In this group of processes, each is assigned a unique rank, and they explicitly communicate with one another by their ranks.

The foundation of communication is built upon send and receive operations among processes. A process may send a message to another process by providing the rank of the process and a unique tag to identify the message. The receiver can then post a receive for a message with a given tag (or it may not even care about the tag), and then handle the data accordingly. Communications such as this which involve one sender and receiver are known as point-to-point communications.

There are many cases where processes may need to communicate with everyone else. For example, when a manager process needs to broadcast information to all of its worker processes. In this case, it would be cumbersome to write code that does all of the sends and receives. In fact, it would often not use the network in an optimal manner. MPI can handle a wide variety of these types of collective communications that involve all processes.

Mixtures of point-to-point and collective communications can be used to create highly complex parallel programs. In fact, this functionality is so powerful that it is not even necessary to start describing the advanced mechanisms of MPI. (…)

*Wes Kendall*

2022

https://mpitutorial.com/tutorials/mpi-introduction/