

## 1 Environnement

### 1.1 Initialisation de l'environnement MPI

```
int MPI_Init(int * argc, char **argv)
```

### 1.2 Rang du processus

```
int MPI_Comm_rank(MPI_Comm comm, int * rang)
```

### 1.3 Nombre de processus

```
int MPI_Comm_size(MPI_Comm comm, int * nb_procs)
```

### 1.4 Désactivation de l'environnement MPI

```
int MPI_Finalize(void)
```

### 1.5 Arrêt d'un programme MPI

```
int MPI_Abort(MPI_Comm comm, int error)
```

### 1.6 Prise de temps

```
double MPI_Wtime(void)
```

## 2 Communications point à point

### 2.1 Envoi de message

```
int MPI_Send(
    const void * message,
    int longueur,
    MPI_Datatype type,
    int rang_dest,
    int etiquette,
    MPI_Comm comm)
```

### 2.2 Envoi non bloquant de message

```
int MPI_Isend(
    const void * message,
    int longueur,
    MPI_Datatype type,
    int rang_dest,
    int etiquette,
    MPI_Comm comm,
    MPI_Request * requete)
```

### 2.3 Réception de message

```
int MPI_Recv(
    void * message,
    int longueur,
    MPI_Datatype type,
    int rang_source,
    int etiquette,
    MPI_Comm comm,
    MPI_Status * statut)
```

### 2.4 Réception non bloquant de message

```
int MPI_Irecv(
    void * message,
    int longueur,
    MPI_Datatype type,
    int rang_source,
    int etiquette,
    MPI_Comm comm,
    MPI_Request * requete)
```

### 2.5 Envoi et réception de message

```
int MPI_Sendrecv(
    cont void * message_emis,
    int longueur_message_emis,
    MPI_Datatype type_message_emis,
    int rang_dest,
    int etiquette_message_emis,
    void * message_recu,
    int longueur_message_recu,
    MPI_Datatype type_message_recu,
    int rang_source,
    int etiquette_message_recu,
    MPI_Comm comm,
    MPI_Status * statut)
```

```
int MPI_Sendrecv_replace(
    void * message_emis_recu,
    int longueur,
    MPI_Datatype type,
    int rang_dest,
    int etiquette_message_emis,
    int rang_source,
    int etiquette_message_recu,
    MPI_Comm comm,
    MPI_Status * statut)
```

### 2.6 Attente de la fin d'une communication non bloquante

```
int MPI_Wait(MPI_Request * requete, MPI_Status * statut)
```

### 2.7 Test de la fin d'une communication non bloquante

```
int MPI_Test(
    MPI_Request * requete,
    int * drapeau,
    MPI_Status * statut)
```

## 3 Communications collectives

### 3.1 Diffusion générale

```
int MPI_Bcast(
    void * message,
    int longueur,
    MPI_Datatype type,
    int rang_source,
    MPI_Comm comm)
```

### 3.2 Diffusion sélective

```
int MPI_Scatter(
    const void * message_a_repartir,
    int longueur_message_emis,
    MPI_Datatype type_message_emis,
    void * message_recu,
    int longueur_message_recu,
    MPI_Datatype type_message_recu,
    int rang_source,
    MPI_Comm comm)
```

### 3.3 Collecte

```
int MPI_Gather(
    const void * message_emis,
    int longueur_message_emis,
    MPI_Datatype type_message_emis,
    void * message_collecte,
    int longueur_message_recu,
    MPI_Datatype type_message_recu,
    int rang_dest,
    MPI_Comm comm)
```

```
int MPI_Allgather(
    const void * message_emis,
    int longueur_message_emis,
    MPI_Datatype type_message_emis,
    void * message_collecte,
    int longueur_message_recu,
    MPI_Datatype type_message_recu,
    MPI_Comm comm)
```

### 3.4 Collecte et diffusion

```
int MPI_Alltoall(
    const void * message_a_repartir,
    int longueur_message_emis,
    MPI_Datatype type_message_emis,
    void * message_collecte,
    int longueur_message_recu,
    MPI_Datatype type_message_recu,
    MPI_Comm comm)
```

## 3.5 Réduction

```
int MPI_Reduce(
    const void * message_emis,
    void * message_recu,
    int longueur,
    MPI_Datatype type,
    MPI_Op operation,
    int rang_dest,
    MPI_Comm comm)

operation ≡ MPI_MAX | MPI_MIN | MPI_SUM | MPI_PROD |
             MPI_BAND | MPI_BOR | MPI_BXOR | MPI_LAND |
             MPI_LOR | MPI_LXOR
```

.....

```
int MPI_Allreduce(
    const void * message_emis,
    void * message_recu,
    int longueur,
    MPI_Datatype type,
    MPI_Op operation,
    MPI_Comm comm)
```

## 3.6 Synchronisation globale

```
int MPI_BARRIER(MPI_Comm comm)
```

## 4 Types dérivés

### 4.1 Types contigus

```
int MPI_Type_contiguous(
    int nb_elements,
    MPI_Datatype ancien_type,
    MPI_Datatype * nouveau_type)
```

### 4.2 Types avec un pas constant

```
int MPI_Type_vector(
    int nombre_bloc,
    int nbr_elt_par_bloc,
    int pas,
    MPI_Datatype type_elt,
    MPI_Datatype * nouveau_type)
```

.....

```
int MPI_Type_create_hvector(
    int nombre_bloc,
    int nbr_elt_par_bloc,
    MPI_Aint pas,
    MPI_Datatype type_elt,
    MPI_Datatype * nouveau_type)
```

### 4.3 Types à pas variable

```
int MPI_Type_indexed(
    int nb_elements,
    const int longueur_bloc[],
    const int pas[],
    MPI_Datatype ancien_type,
    MPI_Datatype * nouveau_type)
```

## 4.4 Types sous-tableau

```
int MPI_Type_create_subarray(
    int nb_dims,
    const int profil_tab[],
    const int profil_sous_tab[],
    const int adresse_debut[],
    int ordre,
    MPI_Datatype ancien_type,
    MPI_Datatype * nouveau_type)
```

## 4.5 Types hétérogènes

```
int MPI_Type_create_struct(
    int nb_elements,
    const int longueur_bloc[],
    const MPI_Aint pas[],
    const MPI_Datatype ancien_types[],
    MPI_Datatype * nouveau_type)
```

## 4.6 Validation des types

```
int MPI_Type_commit(MPI_Datatype * type)
```

## 4.7 Étendue

```
int MPI_Type_get_extent(
    MPI_Datatype type,
    MPI_Aint * borne_inf_alinee,
    MPI_Aint * taille_alinee)
```

.....

```
int MPI_Type_create_resized(
    MPI_Datatype ancien_type,
    MPI_Aint nouvelle_borne_inf,
    MPI_Aint nouvelle_taille,
    MPI_Datatype * nouveau_type)
```

## 4.8 Taille d'un type

```
int MPI_Type_size(MPI_Datatype type, int * taille)
```

## 5 Communicateur

### 5.1 Partitionnement d'un communicateur

```
int MPI_Comm_split(
    MPI_Comm comm,
    int couleur,
    int cle,
    MPI_Comm * nouveau_comm)
```

### 5.2 Création d'une topologie cartésienne

```
int MPI_Cart_create(
    MPI_Comm comm,
    int nb_dims,
    const int dims[],
    const int periodique[],
    int reorganise,
    MPI_Comm * nouveau_comm)
```

## 5.3 Distribution des processus

```
int MPI_Dims_create(
    int nb_procs,
    int nb_dims,
    int dims[])
```

## 5.4 Rang d'un processus

```
int MPI_Cart_rank(
    MPI_Comm comm,
    const int coords[],
    int * rang)
```

## 5.5 Coordonnées d'un processus

```
int MPI_Cart_coords(
    MPI_Comm comm,
    int rang,
    int nb_dims,
    int coords[])
```

## 5.6 Rang des voisins

```
int MPI_Cart_shift(
    MPI_Comm comm,
    int direction,
    int pas,
    int * rang_source,
    int * rang_dest)
```

## 5.7 Subdivision d'une topologie

```
int MPI_Cart_sub(
    MPI_Comm comm,
    const int dim_sub[],
    MPI_Comm * comm_sub)
```

## 6 MPI-IO

### 6.1 Ouverture d'un fichier

```
int MPI_File_open(
    MPI_Comm comm,
    const char * fichier,
    int attribut,
    MPI_Info info,
    MPI_File * descripteur)
```

### 6.2 Fermeture d'un fichier

```
int MPI_File_close(MPI_File * descripteur)
```

### 6.3 Changement de la vue

```
int MPI_File_set_view(
    MPI_File descripteur,
    MPI_Offset déplacement_initial,
    MPI_Datatype type_derive,
    MPI_Datatype motif,
    const char * mode,
    MPI_Info info)
```

## 6.4 Pointeur individuels

### 6.4.1 Positionnement du pointeur

```
int MPI_File_seek(  
    MPI_File    descripteur,  
    MPI_Offset  position_fichier,  
    int         mode_seek)
```

### 6.4.2 Lecture non bloquante

```
int MPI_File_iread(  
    MPI_File    descripteur,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Request * requete)
```

### 6.4.3 Lecture

```
int MPI_File_read(  
    MPI_File    descripteur,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

### 6.4.4 Lecture collective

```
int MPI_File_read_all(  
    MPI_File    descripteur,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

### 6.4.5 Écriture

```
int MPI_File_write(  
    MPI_File    descripteur,  
    const void * valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

### 6.4.6 Écriture collective

```
int MPI_File_write_all(  
    MPI_File    descripteur,  
    const void * valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

## 6.5 Adresses explicites

### 6.5.1 Lecture non bloquante

```
int MPI_File_iread_at(  
    MPI_File    descripteur,  
    MPI_Offset  position_fichier,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Request * requete)
```

### 6.5.2 Lecture

```
int MPI_File_read_at(  
    MPI_File    descripteur,  
    MPI_Offset  position_fichier,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

### 6.5.3 Lecture collective

```
int MPI_File_read_at_all(  
    MPI_File    descripteur,  
    MPI_Offset  position_fichier,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

### 6.5.4 Écriture

```
int MPI_File_write_at(  
    MPI_File    descripteur,  
    MPI_Offset  position_fichier,  
    const void * valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

## 6.6 Pointeurs partagés

### 6.6.1 Positionnement du pointeur

```
int MPI_File_seek_shared(  
    MPI_File    descripteur,  
    MPI_Offset  position_fichier,  
    int         mode_seek)
```

### 6.6.2 Lecture

```
int MPI_File_read_shared(  
    MPI_File    descripteur,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

### 6.6.3 Lecture collective

```
int MPI_File_read_ordered(  
    MPI_File    descripteur,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive,  
    MPI_Status * statut)
```

### 6.6.4 Lecture collective non bloquante

```
int MPI_File_read_ordered_begin(  
    MPI_File    descripteur,  
    void *      valeurs,  
    int         nb_valeurs,  
    MPI_Datatype type_derive)
```

```
int MPI_File_read_ordered_end(  
    MPI_File    descripteur,  
    void *      valeurs,  
    MPI_Status * statut)
```

## 7 Constantes symboliques

```
MPI_ANY_TAG, MPI_ANY_SOURCE, MPI_SUCCESS, MPI_STATUS_IGNORE  
MPI_CHARACTER, MPI_LOGICAL, MPI_INT,  
MPI_FLOAT, MPI_DOUBLE,  
MPI_COMM_NULL, MPI_COMM_WORLD,  
MPI_PROC_NULL, MPI_STATUS_SIZE, MPI_UNDEFINED
```