

1 Environnement

```
integer :: code
MPI_INIT(<OUT> code)

integer :: comm, rang, code
MPI_COMM_RANK(<IN> comm, <OUT> rang, <OUT> code)

integer :: comm, nb_procs, code
MPI_COMM_SIZE(<IN> comm, <OUT> nb_procs, <OUT> code)

integer :: code
MPI_FINALIZE(<OUT> code)

integer :: comm, error, code
MPI_ABORT(<IN> comm, <IN> error, <OUT> code)

real(kind=8) :: temps
temps=MPI_WTIME()
```

2 Communications point à point

```
<type et attribut>:: message
integer :: longueur, type, rang_dest
integer :: etiquette, comm, code
MPI_SEND(
  <IN> message,
  <IN> longueur,
  <IN> type,
  <IN> rang_dest,
  <IN> etiquette,
  <IN> comm,
  <OUT> code)

<type et attribut>:: message
integer :: longueur, type, rang_dest
integer :: etiquette, comm, requete, code
MPI_ISEND(
  <IN> message,
  <IN> longueur,
  <IN> type,
  <IN> rang_dest,
  <IN> etiquette,
  <IN> comm,
  <OUT> requete,
  <OUT> code)

<type et attribut>:: message
integer :: longueur, type, rang_source
integer :: etiquette, comm, code
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_RECV(
  <OUT> message,
  <IN> longueur,
  <IN> type,
  <IN> rang_source,
  <IN> etiquette,
  <IN> comm,
  <OUT> statut,
  <OUT> code)

<type et attribut>:: message
integer :: longueur, type, rang_source
integer :: etiquette, comm, requete, code
MPI_IRECV(
```

```
<OUT> message,
<IN> longueur,
<IN> type,
<IN> rang_source,
<IN> etiquette,
<IN> comm,
<OUT> requete,
<OUT> code)

<type et attribut>:: message_emis, message_recu
integer :: longueur_message_emis, type_message_emis
integer :: etiquette_message_emis, etiquette_message_recu
integer :: longueur_message_recu, type_message_recu
integer :: rang_source, rang_dest, comm, code
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_SENDRECV(
  <IN> message_emis,
  <IN> longueur_message_emis,
  <IN> type_message_emis,
  <IN> rang_dest,
  <IN> etiquette_message_emis,
  <OUT> message_recu,
  <IN> longueur_message_recu,
  <IN> type_message_recu,
  <IN> rang_source,
  <IN> etiquette_message_recu,
  <IN> comm,
  <OUT> statut,
  <OUT> code)

<type et attribut>:: message_emis_recu
integer :: longueur, type, rang_dest, rang_source
integer :: etiquette_message_emis, etiquette_message_recu
integer :: comm, code
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_SENDRECV_REPLACE(
  <INOUT> message_emis_recu,
  <IN> longueur,
  <IN> type,
  <IN> rang_dest,
  <IN> etiquette_message_emis,
  <IN> rang_source,
  <IN> etiquette_message_recu,
  <IN> comm,
  <OUT> statut,
  <OUT> code)

integer :: requete, code
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_WAIT(<IN> requete, <OUT> statut, <OUT> code)
integer :: requete, code

logical :: drapeau
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_TEST(
  <IN> requete,
  <OUT> drapeau,
  <OUT> statut,
  <OUT> code)

integer :: rang_source, etiquette, comm, code
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_PROBE(
  <IN> rang_source,
  <IN> etiquette,
  <IN> comm,
  <OUT> statut,
  <OUT> code)

integer :: requete, code
MPI_REQUEST_FREE(<IN> requete, <OUT> code)
```

3 Communications collectives

```
<type et attribut>:: message
integer :: longueur, type, rang_source, comm, code
MPI_BCAST(
  <INOUT> message,
  <IN> longueur,
  <IN> type,
  <IN> rang_source,
  <IN> comm,
  <OUT> code)

<type et attribut>:: message_emis, message_collecte
integer :: longueur_message_emis, longueur_message_recu
integer :: type_message_emis, type_message_recu
integer :: rang_dest, comm, code
MPI_GATHER(
  <IN> message_emis,
  <IN> longueur_message_emis,
  <IN> type_message_emis,
  <OUT> message_collecte,
  <IN> longueur_message_recu,
  <IN> type_message_recu,
  <IN> rang_dest,
  <IN> comm,
  <OUT> code)

<type et attribut>:: message_a_repartir, message_recu
integer :: longueur_message_emis, longueur_message_recu
integer :: type_message_emis, type_message_recu
integer :: rang_source, comm, code
MPI_SCATTER(
  <IN> message_a_repartir,
  <IN> longueur_message_emis,
  <IN> type_message_emis,
  <OUT> message_recu,
  <IN> longueur_message_recu,
  <IN> type_message_recu,
  <IN> rang_source,
  <IN> comm,
  <OUT> code)

<type et attribut>:: message_a_repartir, message_recu
integer :: longueur_message_emis, longueur_message_recu
integer :: type_message_emis, type_message_recu
integer :: comm, code
MPI_ALLGATHER(
  <IN> message_emis,
  <IN> longueur_message_emis,
  <IN> type_message_emis,
  <OUT> message_collecte,
  <IN> longueur_message_recu,
  <IN> type_message_recu,
  <IN> comm,
  <OUT> code)

<type et attribut>:: message_a_repartir, message_collecte
integer :: longueur_message_emis, longueur_message_recu
integer :: type_message_emis, type_message_recu
integer :: comm, code
MPI_ALLTOALL(
  <IN> message_a_repartir,
  <IN> longueur_message_emis,
  <IN> type_message_emis,
  <OUT> message_collecte,
  <IN> longueur_message_recu,
  <IN> type_message_recu,
  <IN> comm,
  <OUT> code)
```

```

<type et attribut>:: message_emis, message_recu
integer :: longueur, type, rang_dest
integer :: operation, comm, code
MPI_REDUCE(
  <IN>    message_emis,
  <OUT>   message_recu,
  <IN>    longueur,
  <IN>    type,
  <IN>    operation,
  <IN>    rang_dest,
  <IN>    comm,
  <OUT>   code)
operation ≡ MPI_MAX | MPI_MIN | MPI_SUM | MPI_PROD |
MPI_BAND | MPI_BOR | MPI_BXOR | MPI_LAND |
MPI_LOR | MPI_LXOR

```

```

<type et attribut>:: message_emis, message_recu
integer :: longueur, type, operation, comm, code
MPI_ALLREDUCE(
  <IN>    message_emis,
  <OUT>   message_recu,
  <IN>    longueur,
  <IN>    type,
  <IN>    operation,
  <IN>    comm,
  <OUT>   code)

```

```

integer :: comm, code
MPI_BARRIER(<IN> comm, <OUT> code)

```

4 Types dérivés

```

integer :: nb_elements, ancien_type, nouveau_type, code
MPI_TYPE_CONTIGUOUS(
  <IN>    nb_elements,
  <IN>    ancien_type,
  <OUT>   nouveau_type,
  <OUT>   code)

```

```

integer :: nb_elements, longueur_bloc
integer :: pas, ancien_type, nouveau_type, code
MPI_TYPE_VECTOR(
  <IN>    nb_elements,
  <IN>    longueur_bloc,
  <IN>    pas,
  <IN>    ancien_type,
  <OUT>   nouveau_type,
  <OUT>   code)

```

```

integer :: nb_elements, longueur_bloc
integer(MPI_ADDRESS_KIND) :: pas
integer :: ancien_type, nouveau_type, code
MPI_TYPE_CREATE_HVECTOR(
  <IN>    nb_elements,
  <IN>    longueur_bloc,
  <IN>    pas,
  <IN>    ancien_type,
  <OUT>   nouveau_type,
  <OUT>   code)

```

```

integer :: nb_elements, code
integer, dimension(nb_elements) :: longueur_bloc, pas
integer :: ancien_type, nouveau_type
MPI_TYPE_INDEXED(
  <IN>    nb_elements,
  <IN>    longueur_bloc,
  <IN>    pas,
  <IN>    ancien_type,
  <OUT>   nouveau_type,
  <OUT>   code)

```

```

integer :: nb_dims, adresse_debut, ordre
integer :: ancien_type, nouveau_type, code
integer, dimension(nb_dims) :: profil_tab, profil_sous_tab

```

```

integer, dimension(nb_dims) :: adresse_debut
MPI_TYPE_CREATE_SUBARRAY(
  <IN>    nb_dims,
  <IN>    profil_tab,
  <IN>    profil_sous_tab,
  <IN>    adresse_debut,
  <IN>    ordre,
  <IN>    ancien_type,
  <OUT>   nouveau_type,
  <OUT>   code)
integer :: nb_elements, nouveau_type, code
integer, dimension(nb_elements) :: longueur_bloc
integer(MPI_ADDRESS_KIND), dimension(nb_elements) :: pas
integer, dimension(nb_elements) :: ancien_types
MPI_TYPE_CREATE_STRUCT(
  <IN>    nb_elements,
  <IN>    longueur_bloc,
  <IN>    pas,
  <IN>    ancien_types,
  <OUT>   nouveau_type,
  <OUT>   code)

```

```

integer :: type, code
MPI_TYPE_COMMIT(<IN> type, <OUT> code)

```

```

integer :: type, code
MPI_TYPE_FREE(<IN> type, <OUT> code)

```

```

integer :: type, code
integer(MPI_ADDRESS_KIND) :: borne_inf_alignee
integer(MPI_ADDRESS_KIND) :: taille_alignee
MPI_TYPE_GET_EXTENT(
  <IN> type,
  <OUT> borne_inf_alignee,
  <OUT> taille_alignee,
  <OUT> code)

```

```

integer :: ancien_type, nouveau_type, code
integer(MPI_ADDRESS_KIND) :: nouvelle_borne_inf
integer(MPI_ADDRESS_KIND) :: nouvelle_taille
MPI_TYPE_CREATE_RESIZED(
  <IN> ancien_type,
  <IN> nouvelle_borne_inf,
  <IN> nouvelle_taille,
  <OUT> nouveau_type,
  <OUT> code)

```

```

integer :: type, taille, code
MPI_TYPE_SIZE(<IN> type, <OUT> taille, <OUT> code)

```

```

integer :: position, code
integer(MPI_ADDRESS_KIND) :: adresse
MPI_GET_ADDRESS(<IN> position, <OUT> adresse, <OUT> code)

```

5 Communicateur

```

integer :: comm, nb_dims, nouveau_comm, code
integer, dimension(nb_dims) :: dims
logical :: periodique, reorganise
MPI_CART_CREATE(
  <IN>    comm,
  <IN>    nb_dims,
  <IN>    dims,
  <IN>    periodique,
  <IN>    reorganise,
  <OUT>   nouveau_comm,
  <OUT>   code)

```

```

integer :: nb_procs, nb_dims, code
integer, dimension(nb_dims) :: dims
MPI_DIMS_CREATE(
  <IN>    nb_procs,
  <IN>    nb_dims,
  <INOUT> dims,
  <OUT>   code)

```

```

integer :: comm, rang, code, nb_dims
integer, dimension(nb_dims) :: coords
MPI_CART_RANK(
  <IN>    comm,
  <IN>    coords,
  <OUT>   rang,
  <OUT>   code)

```

```

integer :: comm, rang, nb_dims, code
integer, dimension(nb_dims) :: coords
MPI_CART_COORDS(
  <IN>    comm,
  <IN>    rang,
  <IN>    nb_dims,
  <OUT>   coords,
  <OUT>   code)

```

```

integer :: comm, direction, sens
integer :: rang_source, rang_dest, code
MPI_CART_SHIFT(
  <IN>    comm,
  <IN>    direction,
  <IN>    pas,
  <OUT>   rang_source,
  <OUT>   rang_dest,
  <OUT>   code)

```

```

integer :: comm, comm_sub, code, nb_dims
logical, dimension(nb_dims) :: dim_sub
MPI_CART_SUB(
  <IN>    comm,
  <IN>    dim_sub,
  <OUT>   comm_sub,
  <OUT>   code)

```

```

integer :: comm, couleur, cle
integer :: nouveau_comm, code
MPI_COMM_SPLIT(
  <IN>    comm,
  <IN>    couleur,
  <IN>    cle,
  <OUT>   nouveau_comm,
  <OUT>   code)

```

```

integer :: comm, code
MPI_COMM_FREE(<IN> comm, <OUT> code)

```

6 MPI-IO

```

integer :: comm, attribut, info, descripteur, code
character(len=*) :: fichier
MPI_FILE_OPEN(
  <IN>    comm,
  <IN>    fichier,
  <IN>    attribut,
  <IN>    info,
  <OUT>   descripteur,
  <OUT>   code)

```

```

integer :: descripteur, code
MPI_FILE_CLOSE(
  <INOUT> descripteur,
  <OUT>   code)

```

```

integer :: descripteur, type_derive, motif, info, code
integer(kind=MPI_OFFSET_KIND) :: deplacement_initial
character(len=*) :: mode
MPI_FILE_SET_VIEW(
  <INOUT> descripteur,
  <IN>    deplacement_initial,
  <IN>    type_derive,
  <IN>    motif,
  <IN>    mode,
  <IN>    info,
  <OUT>   code)

```

```

integer :: descripteur, mode_seek, code
integer(kind=MPI_OFFSET_KIND) :: position_fichier
MPI_FILE_SEEK(
  <INOUT> descripteur,
  <IN> position_fichier,
  <IN> mode_seek,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, requete
integer :: code
<type et attributs>:: valeurs
MPI_FILE_IREAD(
  <IN> descripteur,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> requete,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_READ(
  <IN> descripteur,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_READ_ALL(
  <IN> descripteur,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_WRITE(
  <IN> descripteur,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut

```

```

MPI_FILE_WRITE_ALL(
  <IN> descripteur,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, requete
integer :: code
integer(kind=MPI_OFFSET_KIND) :: position_fichier
<type et attributs>:: valeurs
MPI_FILE_IREAD_AT(
  <IN> descripteur,
  <IN> position_fichier,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> requete,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
integer(kind=MPI_OFFSET_KIND) :: position_fichier
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_READ_AT(
  <IN> descripteur,
  <IN> position_fichier,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
integer(kind=MPI_OFFSET_KIND) :: position_fichier
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_READ_AT_ALL(
  <IN> descripteur,
  <IN> position_fichier,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
integer(kind=MPI_OFFSET_KIND) :: position_fichier
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_WRITE_AT(
  <IN> descripteur,
  <IN> position_fichier,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

```

```

integer :: descripteur, mode_seek, code
integer(kind=MPI_OFFSET_KIND) :: position_fichier
MPI_FILE_SEEK_SHARED(
  <INOUT> descripteur,
  <IN> position_fichier,
  <IN> mode_seek,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_READ_SHARED(
  <IN> descripteur,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_READ_ORDERED(
  <IN> descripteur,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> statut,
  <OUT> code)

integer :: descripteur, nb_valeurs, type_derive, code
<type et attributs>:: valeurs
MPI_FILE_READ_ORDERED_BEGIN(
  <IN> descripteur,
  <OUT> valeurs,
  <IN> nb_valeurs,
  <IN> type_derive,
  <OUT> code)

integer :: descripteur, code
<type et attributs>:: valeurs
integer, dimension(MPI_STATUS_SIZE) :: statut
MPI_FILE_READ_ORDERED_END(
  <IN> descripteur,
  <OUT> valeurs,
  <OUT> statut,
  <OUT> code)

```

7 Constantes symboliques

```

MPI_ANY_TAG, MPI_ANY_SOURCE, MPI_SUCCESS, MPI_STATUS_IGNORE,
MPI_CHARACTER, MPI_LOGICAL, MPI_INTEGER,
MPI_REAL, MPI_DOUBLE_PRECISION, MPI_COMPLEX,
MPI_COMM_NULL, MPI_COMM_WORLD,
MPI_PROC_NULL, MPI_STATUS_SIZE, MPI_UNDEFINED

```