1 - Environnement

- **Initialiser MPI et quitter MPI :**
  int MPI_Init(int *argc,char*** argv)
  int MPI_Finalize(void)
- **Quitter brutalement MPI :**
  int MPI_Abort(MPI_Comm comm,int errorcode)
- **Savoir si un processus à fait un MPI_Init :**
  int MPI_Initialized(int *flag)
- **Récupérer la chaine de caractères associée au code d'erreur err:**
  int MPI_Error_string(int errcode,char *chaine,int *taille_chaine)
- **Fonctions de timing MPI :**
  doubleMPI_Wtime(void)
  doubleMPI_Wtick(void)

2 - Communications point à point bloquantes

- **Envoyer un message à un processus :**
  int MPI_[R,S,B]send(void *buf,int count,MPI_Datatype datatype,int dest,int tag,MPI_Comm comm)
- **Recevoir un message d'un processus :**
  int MPI_Recv(void *buf,int count,MPI_Datatype datatype,int source,int tag,MPI_Comm comm,MPI_Status *status)
- **Envoyer et recevoir un message :**
  int MPI_Sendrecv(void *sendbuf,int sendcount,MPI_Datatype sendtype,int dest,int sendtag,void *recvbuf,int recvcount,MPI_Datatype recvtype,int source,int recvtag,MPI_Comm comm,MPI_Status *status)
- **Compter le nombre d'éléments reçus :**
  int MPI_Get_count(MPI_status *status,MPI_Datatype datatype,int *count)
- **Tester l'arrivée d'un message :**
  int MPI_Probe(int source,int tag,MPI_Comm comm,MPI_Status *status)
- **Autres fonctions :**
  MPI_Sendrecv_replace(), MPI_Get_elements()

3 - Communications point à point non bloquantes

- **Commencer à envoyer un message :**
  int MPI_I[r,s,b]send(void *buf,int count,MPI_Datatype datatype,int dest,int tag,MPI_Comm comm,MPI_Request * request)
- **Commencer à recevoir un message :**
  int MPI_Irecv(void *buf,int count,MPI_Datatype datatype,int source,int tag,MPI_Comm comm,MPI_Request *request)
- **Compléter une opération non bloquante :**
  int MPI_Wait(MPI_Request *request,MPI_Status *status)
- **Tester une opération non bloquante :**

int MPI_Test(MPI_Request *request,int *flag,MPI_Status *status)

- **Libérer une requête avant de la réutiliser :**
  int MPI_Request_free(MPI_Request *request)
- **autres fonctions :**
  MPI_Waitany(), MPI_Waitall(), MPI_Cancel(), MPI_Testany(), MPI_Iprobe(), MPI_Test_cancelled()

4 - Communications persistantes

- **Décrire un schéma persistant :**
  int MPI_[R,S,B]send_init(void *sendbuf,int count,MPI_Datatype datatype,int dest,int tag,MPI_Comm comm,MPI_Request *request)

  int MPI_Recv_init(void *recbuf,int count,MPI_Datatype datatype,int dest,int tag,MPI_Comm comm,MPI_Request *request)
- **Démarrer une communication persistante :**
  int MPI_Start(MPI_Request *request)
- **Autres fonctions :** MPI_Startall(), MPI_Request_free()

5 - Communications collectives

- **Diffusion générale d'un message :**
  int MPI_Bcast(void *buf,int count,MPI_Datatype datatype,int root,MPI_Comm comm)
- **Collecte de données :**
  int MPI_Gather(void *sendbuf,int sendcount,MPI_Datatype sendtype,void *recvbuf,int recvcount,MPI_Datatype recvtype ,int root,MPI_Comm comm)
- **Diffusion sélective d'un message :**
  int MPI_Scatter(void *sendbuf,int sendcount,MPI_Datatype sendtype,void *recvbuf,int recvcount,MPI_Datatype recvtype ,int root,MPI_Comm comm)
- **Collecte de données et rediffusion :**
  int MPI_Alltoall(void *sendbuf,int sendcount,MPI_Datatype sendtype,void *recvbuf,int recvcount,MPI_Datatype recvtype,MPI_Comm comm)
- **Calcul d'une réduction :**
  int MPI_Reduce(void *sendbuf,void *recvbuf,int count,MPI_Datatype datatype,MPI_Op operation,int root,MPI_Comm comm)
- **Calcul d'une réduction et rediffusion du résultat :**
  int MPI_Allreduce(void *sendbuf,void *recvbuf,int count,MPI_Datatype datatype,MPI_Op operation,MPI_Comm comm)
- **Synchronisation de processus :**
  int MPI_Barrier(MPI_Comm comm)
- **Autres fonctions :**
  MPI_Gatherv(), MPI_Scatterv(),MPI_Allgather[v](), MPI_Alltoallv(), MPI_Op_create(), MPI_Op_free(), MPI_Scan(), MPI_Reduce_scatter()

6 - Communicateurs

- **Création d'un intracommunicateur :**
  int MPI_Comm_create(MPI_Comm comm,MPI_Group group,MPI_Comm *newcomm)
- **Création d'un intercommunicateur :**
  int MPI_Intercomm_create(MPI_Comm comm_local,int leader_local,MPI_Comm comm_lien,int leader_distant,int tag,MPI_Comm *nouvel_intercom)
- **Nombre de processus dans un intracommunicateur :**
  int MPI_Comm_size(MPI_Comm comm,int *nbre)
- **Nombre de processus dans un intercommunicateur :**
  int MPI_Comm_remote_size(MPI_Comm intercomm,int *nbre)
- **Rang d'un processus dans un intracommunicateur :**
  int MPI_Comm_rank(MPI_Comm comm,int *rang)
- **Duplication d'un communicateur :**
  int MPI_Comm_dup(MPI_Comm comm,MPI_Comm *newcomm)
- **Comparaison de deux communicateurs :**
  int MPI_Comm_compare(MPI_Comm comm1,MPI_Comm comm2,int *result)
- **Partage d'un communicateur :**
  int MPI_Comm_split(MPI_Comm comm,int couleur,int clé,MPI_Comm *newcomm)
- **Libérer un communicateur :**
  int MPI_Comm_free(MPI_Comm comm)

7 - Groupes de processus

- **Récupérer le groupe associé à un communicateur :**
  int MPI_Comm_Group(MPI_Comm comm,MPI_Group *groupe)
- **Rang d'un processus dans un groupe :**
  int MPI_Group_rank(MPI_Group groupe,int *rang)
- **Comparer deux groupes de processus :**
  int MPI_Group_compare(MPI_Group groupe1,MPI_Group groupe2,int *result)
- **Fusion de deux groupes :**
  int MPI_Group_union(MPI_Group groupe1,MPI_Group groupe2,MPI_Group *nouveau_groupe)
- **Créer un groupe à partir d'un existant :**
  int MPI_Group_incl(MPI_Group groupe,int n,int *rangs,MPI_Group *nouveau_groupe)
  int MPI_Group_excl(MPI_Group groupe,int n,int *rangs,MPI_Group *nouveau_groupe)
- **Libérer un groupe de processus :**
  int MPI_Group_free(MPI_Group groupe)
- **Autres fonctions :**
  MPI_Group_translate_ranks(), MPI_Group_intersection()

8 - Topologies de processus

- **Création d'une topologie cartesienne :**
  int MPI_Cart_create(MPI_Comm old_comm,int ndims,int *dims,int * periods,int reorder,MPI_Comm *comm_cart)
- **Extraire la topologie associée à un communicateur :**
  int MPI_Cart_get(MPI_Comm comm,int ndims,int *dims,int *periods,int *coords)
- **Faire générer par MPI une décomposition (x,y,z) :**
  int MPI_Dims_create(int npes_total,int ndims,int *dims)
- **Rang d'un processus dans une topologie cartesienne :**
  int MPI_Cart_rank(MPI_Comm comm,int *coords,int *rang)
- **Coordonnées (x,y,z) d'un processus dans la topologie :**
  int MPI_Cart_coords(MPI_Comm comm,int rang,int ndims,int *coords)
- **Décaler une topologie cartésienne (trouver les voisins) :**
  int MPI_Cart_shift(MPI_Comm comm,int direction,int disp,int *rang_source,int *rang_dest)
- **Autres fonctions :**
  MPI_Cart_sub(), MPI_Cart_map(), MPI_Graph_create(), MPI_Graph_neighbors(), MPI_Graph_neighbors_count(), MPI_Graph_get(), MPI_Topo_test()

9 - Types dérivés

- **Construire un type de données contigües :**
  int MPI_Type_contiguous(int nbre,MPI_Datatype ancien_type,MPI_Datatype *nouveau_type)
- **Type de données distantes d'un pas constant :**
  int MPI_Type_[h]vector(int nbre,int taille_bloc,MPI_Aint pas,MPI_Datatype ancien_type,MPI_Datatype *nouveau_type)
- **Type de données distantes d'un pas variable :**
  int MPI_Type_[h]indexed(int nbre,int *taille_bloc,MPI_Aint *pas,MPI_Datatype ancien_type,MPI_Datatype * nouveau_type)
- **Contruire un type structuré :**
  int MPI_Type_struct(int nbre,int *taille_bloc,MPI_Aint *pas,MPI_Datatype *anciens_types,MPI_Datatype *nouveau_type)
- **Valider un type :**
  int MPI_Type_commit(MPI_Datatype *dataype)
- **Routine portable pour retourner l'adresse d'une variable :**
  int MPI_Address(void *variable, MPI_Aint *adresse)
- **Autres fonctions :**
  MPI_Type_free, MPI_Type_extent, MPI_Type_size(), MPI_Type_[u,l]b()

10 - Constantes

- **Jokers :**
  MPI_ANY_TAG, MPI_ANY_SOURCE
- **Datatypes élémentaires :**
  MPI_CHAR, MPI_SHORT, MPI_INT, MPI_LONG, MPI_FLOAT, MPI_DOUBLE, MPI_LONG_DOUBLE, MPI_UNSIGNED, MPI_UNSIGNED_CHAR, MPI_UNSIGNED_SHORT ,MPI_UNSIGNED_LONG, MPI_LOGICAL, MPI_BYTE, MPI_PACKED
- **Constantes reservées :**
  MPI_PROC_NULL, MPI_UNDEFINED
- **Communicateurs reservés :**
  MPI_COMM_WORLD, MPI_COMM_SELF
- **Opérateurs de MPI_Reduce et MPI_Allreduce :**
  MPI_MAX, MPI_MIN, MPI_SUM, MPI_PROD, MPI_BAND, MPI_BOR, MPI_BXOR, MPI_LAND, MPI_LOR, MPI_LXOR
- **Résultat de MPI_Comm_Compare et MPI_Group_Compare :**
  MPI_CONGRUENT, MPI_IDENT, MPI_SIMILAR, MPI_UNEQUAL
- **Chaines de caractères retournées par MPI_Error_string :**

Code	Symptome
MPI_SUCCESS	Pas d'erreurs
MPI_ERR_BUFFER	Pointeur sur zone buffer invalide
MPI_ERR_COUNT	Argument count invalide
MPI_ERR_TYPE	Datatype invalide
MPI_ERR_TAG	Tag de message invalide
MPI_ERR_COMM	Communicateur invalide
MPI_ERR_RANK	Rang de processus invalide
MPI_ERR_REQUEST	Request de communication invalide
MPI_ERR_ROOT	Processus root invalide
MPI_ERR_GROUP	Groupe de processus invalide
MPI_ERR_OP	Opération de réduction impossible
MPI_ERR_TOPOLOGY	Topologie invalide
MPI_ERR_DIMS	Arguments de dimensions invalide
MPI_ERR_ARG	Argument invalide
MPI_ERR_UNKNOWN	Erreur inconnue
MPI_ERR_TRUNCATE	Message tronqué en réception
MPI_ERR_OTHER	Erreur connue non présente dans cette liste
MPI_ERR_INTERN	Erreur interne MPI
MPI_ERR_IN_STATUS	Erreur dans un champ status
MPI_ERR_PENDING	Communication en cours
MPI_ERR_LASTCODE	Dernier code d'erreur