

## MPI/C

### Point-to-Point Communication Routines

```
int MPI_Bsend(void* buf,int count,MPI_Datatype datatype, int dest,int tag,MPI_Comm comm);
int MPI_Bsend_init(void* buf,int count,MPI_Datatype datatype, int dest,int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Buffer_attach(void* buffer,int size);
int MPI_Buffer_detach(void* buffer,int *size);
int MPI_Cancel(MPI_Request *request);
int MPI_Get_count(MPI_Status *status,MPI_Datatype datatype, int *count);
int MPI_Get_elements(MPI_Status *status,MPI_Datatype datatype, int *count);
int MPI_Ibsend(void* buf,int count,MPI_Datatype datatype, int dest,int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Iprobe(int source,int tag,MPI_Comm comm,int *flag, MPI_Status *status);
int MPI_Irecv(void* buf,int count,MPI_Datatype datatype, int source,int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Irsend(void* buf,int count,MPI_Datatype datatype, int dest,int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Isend(void* buf,int count,MPI_Datatype datatype,int dest, int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Issend(void* buf,int count,MPI_Datatype datatype,int dest, int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Probe(int source,int tag,MPI_Comm comm,MPI_Status *status);
int MPI_Recv(void* buf,int count,MPI_Datatype datatype, int source,int tag,MPI_Comm comm,MPI_Status *status);
int MPI_Recv_init(void* buf,int count,MPI_Datatype datatype, int source,int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Request_free(int MPI_Request *request);
int MPI_Rsend(void* buf,int count,MPI_Datatype datatype, int dest,int tag,MPI_Comm comm);
int MPI_Rsend_init(void* buf,int count,MPI_Datatype datatype,int dest, int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Send(void* buf,int count,MPI_Datatype datatype, int dest,int tag,MPI_Comm comm);
int MPI_Send_init(void* buf,int count,MPI_Datatype datatype, int dest,int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Sendrecv(void *sendbuf,int sendcount,MPI_Datatype sendtype, int dest,int sendtag,void *recvbuf,int recvcnt,
MPI_Datatype recvtype,int source,int recvtag, MPI_Comm comm,MPI_Status *status);
int MPI_Sendrecv_replace(void* buf,int count,MPI_Datatype datatype, int dest,int sendtag,int source,int recvtag,
MPI_Comm comm,MPI_Status *status);
int MPI_Ssend(void* buf,int count,MPI_Datatype datatype, int dest,int tag,MPI_Comm comm);
int MPI_Ssend_init(void* buf,int count,MPI_Datatype datatype,int dest, int tag,MPI_Comm comm,MPI_Request *request);
int MPI_Start(MPI_Request *request);
int MPI_Startall(int count,MPI_request *array_of_requests);
int MPI_Test(MPI_Request *request,int *flag,MPI_Status *status);
int MPI_Test_cancelled(MPI_Status * status,int *flag);
int MPI_Testall(int count,MPI_Request *array_of_requests, int *flag,MPI_Status *array_of_statuses);
int MPI_Testany(int count,MPI_Request *array_of_requests,int *index, int *flag,MPI_Status *status);
int MPI_Testsome(int incount,MPI_Request *array_of_requests, int *outcount,int *array_of_indices, MPI_Status *array_of_statuses);
int MPI_Wait(MPI_Request *request,MPI_Status *status);
int MPI_Waitall(int count,MPI_Request *array_of_requests, MPI_Status *array_of_statuses);
int MPI_Waitany(int count,MPI_Request *array_of_requests, int *index,MPI_Status *status);
int MPI_Waitsome(int incount,MPI_Request *array_of_requests, int *outcount,int *array_of_indices,MPI_Status *array_of_statuses);
```

### Collective Communication Routines

```
int MPI_Allgather(void* sendbuf,int sendcount,MPI_Datatype sendtype,void* recvbuf,int recvcnt,MPI_Datatype recvtype,MPI_Comm comm);
int MPI_Allgatherv(void* sendbuf,int sendcount,MPI_Datatype sendtype,
void* recvbuf,int *recvcnts,int *displs,MPI_Datatype recvtype, MPI_Comm comm);
int MPI_Allreduce(void* sendbuf,void* recvbuf,int count, MPI_Datatype datatype,MPI_Op op,MPI_Comm comm);
int MPI_Alltoall(void* sendbuf,int sendcount,MPI_Datatype sendtype, void* recvbuf,int recvcnt,MPI_Datatype recvtype,MPI_Comm comm);
int MPI_Alltoallv(void* sendbuf,int *sendcounts,int *sdispls, MPI_Datatype sendtype,void* recvbuf,int *recvcnts,int *rdispls,
MPI_Datatype recvtype,MPI_Comm comm);
int MPI_Barrier(MPI_Comm comm);
int MPI_Bcast(void* buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm);
int MPI_Gather(void* sendbuf,int sendcount,MPI_Datatype sendtype, void* recvbuf,int recvcnt,MPI_Datatype recvtype,int root,
MPI_Comm comm);
int MPI_Gatherv(void* sendbuf,int sendcount,MPI_Datatype sendtype, void* recvbuf,int recvcnts,int *displs,MPI_Datatype recvtype,
int root,MPI_Comm comm);
int MPI_Op_create(MPI_User_function *function,int commute,MPI_Op *op);
int MPI_Op_free(MPI_Op *op);
int MPI_Reduce(void* sendbuf,void* recvbuf,int count, MPI_Datatype datatype,MPI_Op op,int root,MPI_Comm comm);
int MPI_Reduce_scatter(void* sendbuf,void* recvbuf,int *recvcnts, MPI_Datatype datatype,MPI_Op op,MPI_Comm comm);
int MPI_Scan(void* sendbuf,void* recvbuf,int count, MPI_Datatype datatype,MPI_Op op,MPI_Comm comm);
int MPI_Scatter(void* sendbuf,int sendcount,MPI_Datatype sendtype,void* recvbuf,
int recvcnt,MPI_Datatype recvtype,int root,MPI_Comm comm);
int MPI_Scatterv(void* sendbuf,int *sendcounts,int *displs, MPI_Datatype sendtype,void* recvbuf,int recvcnt,
MPI_Datatype recvtype,int root,MPI_Comm comm);
```

### Process Group Routines

```
int MPI_Group_size(MPI_Group group,int *size);
int MPI_Group_translate_ranks(MPI_Group group1,int n, int *ranks1,MPI_Group group2,int *ranks2);
int MPI_Group_union(MPI_Group group1,MPI_Group group2, MPI_Group *newgroup);
int MPI_Group_rank(MPI_Group group,int *rank);
int MPI_Group_range_incl(MPI_Group group,int n, int ranges[][3],MPI_Group *newgroup);
int MPI_Group_range_excl(MPI_Group group,int n, int ranges[][3],MPI_Group *newgroup);
int MPI_Group_free(MPI_Group *group);
int MPI_Group_incl(MPI_Group group,int n,int *ranks, MPI_Group *newgroup);
int MPI_Group_intersection(MPI_Group group1,MPI_Group group2, MPI_Group *newgroup);
int MPI_Group_excl(MPI_Group group,int n,int *ranks, MPI_Group *newgroup);
int MPI_Group_difference(MPI_Group group1,MPI_Group group2, MPI_Group *newgroup);
int MPI_Group_compare(MPI_Group group1,MPI_Group group2, int *result);
```

### struct Status

```
typedef struct MPI_Status {
int count;
int cancelled;
int MPI_SOURCE;
int MPI_TAG;
int MPI_ERROR;
} MPI_Status;
```

## Communicators Routines

```
int MPI_Comm_compare(MPI_Comm comm1, MPI_Comm comm2, int *result);
int MPI_Comm_create(MPI_Comm comm_in, MPI_Group group, MPI_Comm *comm_out);
int MPI_Comm_dup(MPI_Comm comm, MPI_Comm *newcomm);
int MPI_Comm_free(MPI_Comm *comm);
int MPI_Comm_group(MPI_Comm comm, MPI_Group *group);
int MPI_Comm_rank(MPI_Comm comm, int *rank);
int MPI_Comm_remote_group(MPI_Comm comm, MPI_Group *group);
int MPI_Comm_remote_size(MPI_Comm comm, int *size);
int MPI_Comm_size(MPI_Comm comm, int *size);
int MPI_Comm_split(MPI_Comm comm_in, int color, int key, MPI_Comm *comm_out);
int MPI_Comm_test_inter(MPI_Comm comm, int *flag);
int MPI_Intercomm_create(MPI_Comm local_comm, int local_leader, MPI_Comm peer_comm, int remote_leader, int tag, MPI_Comm *newintercom);
int MPI_Intercomm_merge(MPI_Comm intercomm, int high, MPI_Comm *newintracomm);
```

## Derived Types Routines

```
int MPI_Type_commit(MPI_Datatype *datatype);
int MPI_Type_contiguous(int count, MPI_Datatype oldtype, MPI_Datatype *newtype);
int MPI_Type_extent(MPI_Datatype datatype, MPI_Aint *size);
int MPI_Type_free(MPI_Datatype *datatype);
int MPI_Type_hindexed(int count, int *array_of_blocklengths, MPI_Aint *array_of_displacements,
MPI_Datatype oldtype, MPI_Datatype *newtype);
int MPI_Type_hvector(int count, int blocklength, MPI_Aint stride, MPI_Datatype oldtype, MPI_Datatype *newtype);
int MPI_Type_indexed(int count, int *array_of_blocklengths, int *array_of_displacements, MPI_Datatype oldtype, MPI_Datatype *newtype);
int MPI_Type_lb(MPI_Datatype datatype, MPI_Aint *displacement);
int MPI_Type_size(MPI_Datatype datatype, int *size);
int MPI_Type_struct(int count, int *array_of_blocklengths, MPI_Aint *array_of_displacements, MPI_Datatype *array_of_types,
MPI_Datatype *newtype);
int MPI_Type_ub(MPI_Datatype datatype, MPI_Aint *displacement);
int MPI_Type_vector(int count, int blocklength, int stride, MPI_Datatype oldtype, MPI_Datatype *newtype);
```

## Basic Types

MPI\_CHAR, MPI\_SIGNED\_CHAR, MPI\_UNSIGNED\_CHAR, MPI\_BYTE, MPI\_WCHAR, MPI\_SHORT, MPI\_UNSIGNED\_SHORT, MPI\_INT, MPI\_UNSIGNED, MPI\_LONG, MPI\_UNSIGNED\_LONG, MPI\_FLOAT, MPI\_DOUBLE, MPI\_LONG\_DOUBLE, MPI\_LONG\_LONG\_INT, MPI\_UNSIGNED\_LONG\_LONG, MPI\_LONG\_LONG, MPI\_LONG\_LONG\_INT, MPI\_PACKED, MPI\_LB, MPI\_UB

## Operators

MPI\_MAX, MPI\_MIN, MPI\_SUM, MPI\_PROD, MPI\_LAND, MPI\_BAND, MPI\_LOR, MPI\_BOR, MPI\_LXOR, MPI\_BXOR, MPI\_MINLOC, MPI\_MAXLOC, MPI\_REPLACE, MPI\_OP\_NULL

## Virtual Topology Routines

```
MPI_Cart_coords(MPI_Comm comm, int rank, int maxdims, int *coords);
int MPI_Cart_create(MPI_Comm comm_old, int ndims, int *dims, int *periods, int reorder, MPI_Comm *comm_cart);
MPI_Cart_get(MPI_Comm comm, int maxdims, int *dims, int *periods, int *coords);
MPI_Cart_map(MPI_Comm comm, int ndims, int *dims, int *periods, int *newrank);
MPI_Cart_rank(MPI_Comm comm, int *coords, int *rank);
MPI_Cart_shift(MPI_Comm comm, int direction, int disp, int *rank_source, int *rank_dest);
MPI_Cart_sub(MPI_Comm comm, int *remain_dims, MPI_Comm *newcomm);
MPI_Cartdim_get(MPI_Comm comm, int *ndims);
MPI_Dims_create(int nnodes, int ndims, int *dims);
MPI_Graph_create(MPI_Comm comm_old, int nnodes, int *index, int *edges, int reorder, MPI_Comm *comm_graph);
MPI_Graph_get(MPI_Comm comm, int maxindex, int maxedges, int *index, int *edges);
MPI_Graph_map(MPI_Comm comm, int nnodes, int *index, int *edges, int *newrank);
MPI_Graph_neighbors(MPI_Comm comm, int rank, int maxneighbors, int *neighbors);
MPI_Graph_neighbors_count(MPI_Comm comm, int rank, int *neighbors);
MPI_Graphdims_get(MPI_Comm comm, int *nnodes, int *nedges);
MPI_Topo_test(MPI_Comm comm, int *status);
```

## Environment Management Routines

```
int MPI_Abort(MPI_Comm comm, int errorcode);
int MPI_Errhandler_get(MPI_Comm comm, MPI_Errhandler *errhandler);
int MPI_Error_string(int errorcode, char *string, int *resultlen);
int MPI_Init(int *argc, char ***argv);
double MPI_Wtime(void);
int MPI_Errhandler_create(MPI_Handler_function *function, MPI_Errhandler *errhandler);
int MPI_Errhandler_set(MPI_Comm comm, MPI_Errhandler errhandler);
int MPI_Finalize(void);
int MPI_Initialized(int *flag);
int MPI_Errhandler_free(MPI_Errhandler *errhandler);
int MPI_Error_class(int errorcode, int *errorclass);
int MPI_Get_processor_name(char *name, int *resultlen);
double MPI_Wtick(void);
```

## Miscellaneous Routines

```
int MPI_Address(void* location, MPI_Aint *address);
int MPI_Attr_delete(MPI_Comm comm, int keyval);
int MPI_Attr_get(MPI_Comm comm, int keyval, void *attribute_val, int *flag);
int MPI_Attr_put(MPI_Comm comm, int keyval, void* attribute_val);
int MPI_Keyval_create(MPI_Copy_function *copy_fn, MPI_Delete_function *delete_fn, int *keyval, void* extra_state);
int MPI_Keyval_free(int *keyval);
int MPI_Pack(void* inbuf, int incout, MPI_Datatype datatype, void *outbuf, int outsize, int *position, MPI_Comm comm);
int MPI_Pack_size(int incout, MPI_Datatype datatype, MPI_Comm comm, int *size);
int MPI_Pcontrol(const int level, ...);
int MPI_Unpack(void* inbuf, int insize, int *position, void *outbuf, int outcount, MPI_Datatype datatype, MPI_Comm comm);
```