



## Message Passing Interface

### Quick Reference in C

```
#include <mpi.h>
```

#### Blocking Point-to-Point

Send a message to one process. (§3.2.1)

```
int MPI_Send (void *buf, int count,
             MPI_Datatype datatype, int dest, int
             tag, MPI_Comm comm)
```

Receive a message from one process. (§3.2.4)

```
int MPI_Recv (void *buf, int count,
             MPI_Datatype datatype, int source, int
             tag, MPI_Comm comm, MPI_Status *status)
```

Count received data elements. (§3.2.5)

```
int MPI_Get_count (MPI_Status *status,
                 MPI_Datatype datatype, int *count)
```

Wait for message arrival. (§3.8)

```
int MPI_Probe (int source, int tag,
             MPI_Comm comm, MPI_Status *status)
```

*Related Functions:* MPI\_Bsend, MPI\_Ssend, MPI\_Rsend,  
MPI\_Buffer\_attach, MPI\_Buffer\_detach, MPI\_Sendrecv,  
MPI\_Sendrecv\_replace, MPI\_Get\_elements

#### Non-blocking Point-to-Point

Begin to receive a message. (§3.7.2)

```
int MPI_Irecv (void *buf, int count,
             MPI_Datatype, int source, int tag,
             MPI_Comm comm, MPI_Request *request)
```

Complete a non-blocking operation. (§3.7.3)

```
int MPI_Wait (MPI_Request *request,
             MPI_Status *status)
```

Check or complete a non-blocking operation. (§3.7.3)

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

Check message arrival. (§3.8)

```
int MPI_Iprobe (int source, int tag,
             MPI_Comm comm, int *flag, MPI_Status
             *status)
```

*Related Functions:* MPI\_Isend, MPI\_Ibsend, MPI\_Issend,  
MPI\_Irsend, MPI\_Request\_free, MPI\_Waitany,  
MPI\_Testany, MPI\_Waitall, MPI\_Testall, MPI\_Waitsome,  
MPI\_Testsome, MPI\_Cancel, MPI\_Test\_cancelled

#### Persistent Requests

*Related Functions:* MPI\_Send\_init, MPI\_Bsend\_init,  
MPI\_Ssend\_init, MPI\_Rsend\_init, MPI\_Recv\_init,  
MPI\_Start, MPI\_Startall

#### Derived Datatypes

Create a strided homogeneous vector. (§3.12.1)

```
int MPI_Type_vector (int count, int
                 blocklength, int stride, MPI_Datatype
                 oldtype, MPI_Datatype *newtype)
```

Save a derived datatype (§3.12.4)

```
int MPI_Type_commit (MPI_Datatype
                 *datatype)
```

Pack data into a message buffer. (§3.13)

```
int MPI_Pack (void *inbuf, int incount,
             MPI_Datatype datatype, void *outbuf,
             int outsize, int *position, MPI_Comm
             comm)
```

Unpack data from a message buffer. (§3.13)

```
int MPI_Unpack (void *inbuf, int insize,
             int *position, void *outbuf, int
             outcount, MPI_Datatype datatype,
             MPI_Comm comm)
```

Determine buffer size for packed data. (§3.13)

```
int MPI_Pack_size (int incount,
                 MPI_Datatype datatype, MPI_Comm comm,
                 int *size)
```

*Related Functions:* MPI\_Type\_contiguous,  
MPI\_Type\_hvector, MPI\_Type\_indexed,  
MPI\_Type\_hindexed, MPI\_Type\_struct, MPI\_Address,  
MPI\_Type\_extent, MPI\_Type\_size, MPI\_Type\_lb,  
MPI\_Type\_ub, MPI\_Type\_free

#### Collective

Send one message to all group members. (§4.4)

```
int MPI_Bcast (void *buf, int count,
             MPI_Datatype datatype, int root,
             MPI_Comm comm)
```

Receive from all group members. (§4.5)

```
int MPI_Gather (void *sendbuf, int
             sendcount, MPI_Datatype sendtype, void
             *recvbuf, int recvcount, MPI_Datatype
             recvtpe, int root, MPI_Comm comm)
```

Send separate messages to all group members. (§4.6)

```
int MPI_Scatter (void *sendbuf, int
             sendcount, MPI_Datatype sendtype, void
             *recvbuf, int recvcount, MPI_Datatype
             recvtpe, int root, MPI_Comm comm)
```

Combine messages from all group members. (§4.9.1)

```
int MPI_Reduce (void *sendbuf, void
             *recvbuf, int count, MPI_Datatype
             datatype, MPI_Op op, int root, MPI_Comm
             comm)
```

*Related Functions:* MPI\_Barrier, MPI\_Gatherv,  
MPI\_Scatterv, MPI\_Allgather, MPI\_Allgatherv,  
MPI\_Alltoall, MPI\_Alltoallv, MPI\_Op\_create,  
MPI\_Op\_free, MPI\_Allreduce, MPI\_Reduce\_scatter,  
MPI\_Scan

#### Groups

*Related Functions:* MPI\_Group\_size, MPI\_Group\_rank,  
MPI\_Group\_translate\_ranks, MPI\_Group\_compare,  
MPI\_Comm\_group, MPI\_Group\_union,  
MPI\_Group\_intersection, MPI\_Group\_difference,  
MPI\_Group\_incl, MPI\_Group\_excl,  
MPI\_Group\_range\_incl, MPI\_Group\_range\_excl,  
MPI\_Group\_free

#### Basic Communicators

Count group members in communicator. (§5.4.1)

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

Determine group rank of self. (§5.4.1)

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

Duplicate with new context. (§5.4.2)

```
int MPI_Comm_dup (MPI_Comm comm, MPI_Comm
                 *newcomm)
```

Split into categorized sub-groups. (§5.4.2)

```
int MPI_Comm_split (MPI_Comm comm, int
                 color, int key, MPI_Comm *newcomm)
```

*Related Functions:* MPI\_Comm\_compare,  
MPI\_Comm\_create, MPI\_Comm\_free,

MPI\_Comm\_test\_inter, MPI\_Comm\_remote\_size,  
MPI\_Comm\_remote\_group, MPI\_Intercomm\_create,  
MPI\_Intercomm\_merge

## Communicators with Topology

Create with cartesian topology. (§6.5.1)

```
int MPI_Cart_create (MPI_Comm comm_old,  
    int ndims, int *dims, int *periods, int  
    reorder, MPI_Comm *comm_cart)
```

Suggest balanced dimension ranges. (§6.5.2)

```
int MPI_Dims_create (int nnodes, int  
    ndims, int *dims)
```

Determine rank from cartesian coordinates. (§6.5.4)

```
int MPI_Cart_rank (MPI_Comm comm, int  
    *coords, int *rank)
```

Determine cartesian coordinates from rank. (§6.5.4)

```
int MPI_Cart_coords (MPI_Comm comm, int  
    rank, int maxdims, int *coords)
```

Determine ranks for cartesian shift. (§6.5.5)

```
int MPI_Cart_shift (MPI_Comm comm, int  
    direction, int disp, int *rank_source,  
    int *rank_dest)
```

Split into lower dimensional sub-grids. (§6.5.6)

```
int MPI_Cart_sub (MPI_Comm comm, int  
    *remain_dims, MPI_Comm *newcomm)
```

*Related Functions:* MPI\_Graph\_create, MPI\_Topo\_test,  
MPI\_Graphdims\_get, MPI\_Graph\_get,  
MPI\_Cartdim\_get, MPI\_Cart\_get,  
MPI\_Graph\_neighbors\_count, MPI\_Graph\_neighbors,  
MPI\_Cart\_map, MPI\_Graph\_map

## Communicator Caches

*Related Functions:* MPI\_Keyval\_create, MPI\_Keyval\_free,  
MPI\_Attr\_put, MPI\_Attr\_get, MPI\_Attr\_delete

## LAM & MPI Information

1224 Kinnear Rd.  
Columbus, Ohio 43212  
614-292-8492

lam@tbag.osc.edu

<http://www.osc.edu/lam.html>  
<ftp://ftp.osc.edu/pub/lam>



## Error Handling

*Related Functions:* MPI\_Errhandler\_create,  
MPI\_Errhandler\_set, MPI\_Errhandler\_get,  
MPI\_Errhandler\_free, MPI\_Error\_string, MPI\_Error\_class

## Dynamic Processes

Spawn a process. (MPI-2)

```
int MPI_Spawn (char prog[], char *argv[],  
    int maxprocs, MPI_Info info, int root,  
    MPI_Comm parents, MPI_Comm *children,  
    int errs[]);
```

*Related Functions:* MPI\_Spawn\_multiple, MPI\_Ispawn,  
MPI\_Ispawn\_multiple, MPI\_Port\_open, MPI\_Port\_Close,  
MPI\_Accept, MPI\_Connect, MPI\_Name\_publish,  
MPI\_Name\_unpublish, MPI\_Name\_get, MPI\_Iaccept,  
MPI\_Iconnect, MPI\_Info\_create, MPI\_Info\_set,  
MPI\_Info\_get, MPI\_Info\_get\_valuelen,  
MPI\_Info\_get\_nkeys, MPI\_Info\_get\_nthkey,  
MPI\_Info\_dup, MPI\_Info\_free, MPI\_Info\_delete

## Environmental

Determine wall clock time. (§7.4)

```
double MPI_Wtime (void)
```

Initialize MPI. (§7.5)

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

Cleanup MPI. (§7.5)

```
int MPI_Finalize (void)
```

*Related Functions:* MPI\_Get\_processor\_name, MPI\_Wtick,  
MPI\_Initialized, MPI\_Abort, MPI\_Pcontrol, MPI\_Get\_version

## Constants

Wildcards (§3.2.4)

```
MPI_ANY_TAG, MPI_ANY_SOURCE
```

Elementary Datatypes (§3.2.2)

```
MPI_CHAR, MPI_SHORT, MPI_INT, MPI_LONG,  
MPI_UNSIGNED_CHAR, MPI_UNSIGNED_SHORT,  
MPI_UNSIGNED, MPI_UNSIGNED_LONG, MPI_FLOAT,  
MPI_DOUBLE, MPI_LONG_DOUBLE, MPI_BYTE,  
MPI_PACKED
```

Reserved Communicators (§5.2.4)

```
MPI_COMM_WORLD, MPI_COMM_SELF, MPI_COMM_PARENT
```

Reduction Operations (§4.9.2)

```
MPI_MAX, MPI_MIN, MPI_SUM, MPI_PROD,  
MPI_BAND, MPI_BOR, MPI_BXOR, MPI_LAND,  
MPI_LOR, MPI_LXOR
```



## LAM Quick Reference

### Session Management

Confirm a group of hosts.  
recon -v <hostfile>

Start LAM on a group of hosts.  
lamboot -v <hostfile>

Terminate LAM.

```
wipe -v <hostfile>
```

Hostfile Syntax

```
# comment  
<hostname> <userid>  
<hostname> <userid>  
...etc...
```

### Compilation

Compile a program for LAM / MPI.

```
hcc -o <binary> <source> -I<incdir>  
-L<libdir> -l<lib> -lmpi
```

### Processes and Messages

Start an SPMD application.

```
mpirun -v -s <src_node> -c <copies>  
<nodes> <program> -- <args>
```

Start a MIMD application.

```
mpirun -v <appfile>
```

Appfile Syntax

```
# comment  
<program> -s <src_node> <nodes> -- <args>  
<program> -s <src_node> <nodes> -- <args>  
...etc...
```

Examine the state of processes.

```
mpitask
```

Examine the state of messages.

```
mpimsg
```

Cleanup all processes and messages.

```
lamclean -v
```