Distribution Category:
Mathematics and
Computer Science (UC-405)

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, IL 60439

ANL-00

MPICH2 Model MPI Implementation Reference Manual

Draft

by

William Gropp
Ewing Lusk Mathematics and Computer Science Division
Argonne National Laboratory

December 12, 2003

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Contents

1 Introduction

This document contains detailed documentation on the routines that are part of the MPICH model MPI implementation.

As an alternate to this manual, the reader should consider using the script `mpiman`; this is a script that uses `xman` to provide a X11 Window System interface to the data in this manual.

2 MPI Commands

MPI **MPI**

MPI — Introduction to the Message-Passing Interface

Description

MPI stands for Message Passing Interface. , simply
MPI is a specification (like C or Fortran) and there are a number of implementations. The rest of this man page describes the use of the MPICH implementation of MPI.

Getting started

Add MPI to your path

```
% set path = ( $path /usr/local/mpi/bin )
```

Compute pi to a given resolution on 8 processors or threads

```
% mpiexec -n 8 /usr/local/mpi/examples/cpi
```

You can compile and link your own MPI programs with the commands `mpicc`, `mpif77`, and `mpicxx`:

```
% mpicc -o cpi cpi.c
% mpif77 -o fpi fpi.f
% mpicxx -o cxxpi cxxpi.cxx
```

Documentation

Postscript documentation can be found in directory /usr/local/mpi/doc/. These include an introductory guide (guide.ps) and a user manual (manual.ps).
Man pages exist for every MPI subroutine and function. The man pages are also available on the Web at `http://www.mcs.anl.gov/mpi/www`. Additional on-line information is available at `http://www.mcs.anl.gov/mpi`, including a hypertext version of the standard, information on other libraries that use MPI, and pointers to other MPI resources.

Version

MPICH2 version 0.92

License

Copyright 20028 University of Chicago See COPYRIGHT for details. The source code is freely available by anonymous ftp from ftp.mcs.anl.gov in pub/mpi/mpich2-beta.tar.gz .

Files

`/usr/local/mpi/`	`MPI software directory`
`/usr/local/mpi/COPYRIGHT`	`Copyright notice`
`/usr/local/mpi/README`	`various notes and instructions`
`/usr/local/mpi/bin/`	`binaries, including mpiexec and mpicc`
`/usr/local/mpi/examples`	`elementary MPI programs`
`/usr/local/mpi/doc/`	`documentation`
`/usr/local/mpi/include/`	`include files`
`/usr/local/mpi/lib/`	`library files`

Contact

For comments regarding the local installation of MPI, please send mail to support@mcs.anl.gov. MPI-specific suggestions and bug reports should be sent directly to mpi-bugs@mcs.anl.gov.

Location

`manpage.txt`

---

mpicc **mpicc**

---

mpicc — Compiles and links MPI programs written in C

Description

This command can be used to compile and link MPI programs written in C. It provides the options and any special libraries that are needed to compile and link MPI programs.
It is important to use this command, particularly when linking programs, as it provides the necessary libraries.

Command line arguments

-show Show the commands that would be used without runnning them

-help Give short help

-cc=name Use compiler `name` instead of the default choice. Use this only if the compiler is compatible with the MPICH library (see below)

-config=name Load a configuration file for a particular compiler. This allows a single `mpicc` command to be used with multiple compilers.

-compile_info Show the steps for compiling a program. This option can be used to see what options and include paths are used by mpicc.

-link_info Show the steps for linking a program. This optoin can be used to see what options and libraries are used by mpicc.

-echo Show exactly what this program is doing. This option should normally not be used.

others are passed to the compiler or linker. For example, `-c` causes files to be compiled, `-g` selects compilation with debugging on most systems, and `-o name` causes linking with the output executable given the name `name`.

Environment Variables

The environment variables `MPICH_CC` may be used to select different C compiler and linker. Note that since MPICH is built with a particular C and Fortran compiler, changing the compilers used can cause problems. Use this only if you could intermix code compiled with the different compilers.

Compatible Compilers

The MPI library may be used with any compiler that uses the same lengths for basic data objects (such as `long double`) and that uses compatible run-time libraries. On many systems, the various compilers are compatible and may be used interchangable. There are exceptions; if you use the `MPICH_CC` environment variable or the `-cc=name` command-line argument to override the choice of compiler and encounter problems, try reconfiguring MPICH2 with the new compiler, installing MPICH2 in a separate location. See the installation manual for more details.

Examples

To compile a single file `foo.c`, use

```
mpicc -c foo.c
```

To link the output and make an executable, use

```
mpicc -o foo foo.o
```

Combining compilation and linking in a single command

```
mpicc -o foo foo.c
```

is a convenient way to build simple programs.

See Also

mpif77, mpicxx, mpif90, mpiexec

Location

`mpicc.txt`

---

mpicxx **mpicxx**

---

mpicxx — Compiles and links MPI programs written in C++

Description

This command can be used to compile and link MPI programs written in C++. It provides the options and any special libraries that are needed to compile and link MPI programs.
It is important to use this command, particularly when linking programs, as it provides the necessary libraries.

Command line arguments

-show Show the commands that would be used without runnning them

-help Give short help

-cxx=name Use compiler `name` instead of the default choice. Use this only if the compiler is compatible with the MPICH library (see below)

-config=name Load a configuration file for a particular compiler. This allows a single `mpicxx` command to be used with multiple compilers.

-compile_info Show the steps for compiling a program. This option can be used to see what options and include paths are used by mpicxx.

-link_info Show the steps for linking a program. This optoin can be used to see what options and libraries are used by mpicxx.

-echo Show exactly what this program is doing. This option should normally not be used.

others are passed to the compiler or linker. For example, `-c` causes files to be compiled, `-g` selects compilation with debugging on most systems, and `-o name` causes linking with the output executable given the name `name`.

Environment Variables

The environment variables `MPICH_CXX` may be used to select different C++ compiler and linker. Note that since MPICH is built with a particular C and Fortran compiler, changing the compilers used can cause problems. Use this only if you could intermix code compiled with the different compilers.

Compatible Compilers

The MPI library may be used with any compiler that uses the same lengths for basic data objects (such as `long double`) and that uses compatible run-time libraries. On many systems, the various compilers are compatible and may be used interchangable. There are exceptions; if you use the `MPICH_CXX` environment variable or the `-cxx=name` command-line argument to override the choice of compiler and encounter problems, try reconfiguring MPICH2 with the new compiler, installing MPICH2 in a separate location. See the installation manual for more details.

Examples

To compile a single file `foo.c`, use

```
mpicxx -c foo.cxx
```

To link the output and make an executable, use

```
mpicxx -o foo foo.o
```

Combining compilation and linking in a single command

```
mpicxx -o foo foo.cxx
```

is a convenient way to build simple programs.

See Also

mpif77, mpicxx, mpif90, mpiexec

Location

`mpicxx.txt`

---

mpif77 **mpif77**

---

mpif77 — Compiles and links MPI programs written in Fortran 77

Description

This command can be used to compile and link MPI programs written in Fortran. It provides the options and any special libraries that are needed to compile and link MPI programs.
It is important to use this command, particularly when linking programs, as it provides the necessary libraries.

Command line arguments

-show Show the commands that would be used without runnning them

-help Give short help

-f77=name Use compiler `name` instead of the default choice. Use this only if the compiler is compatible with the MPICH library (see below)

-config=name Load a configuration file for a particular compiler. This allows a single `mpif77` command to be used with multiple compilers.

-compile_info Show the steps for compiling a program. This option can be used to see what options and include paths are used by mpif77.

-link_info Show the steps for linking a program. This optoin can be used to see what options and libraries are used by mpif77.

-echo Show exactly what this program is doing. This option should normally not be used.

others are passed to the compiler or linker. For example, `-c` causes files to be compiled, `-g` selects compilation with debugging on most systems, and `-o name` causes linking with the output executable given the name `name`.

Environment Variables

The environment variables `MPICH_F77` may be used to select different Fortran compiler and linker. Note that since MPICH is built with a particular C and Fortran compiler, change the compilers used can cause problems. Use this only if you could intermix code compiled with the different compilers.

Compatible Compilers

The MPI library may be used with any compiler that uses the same lengths for basic data objects (such as `long double`) and that uses compatible run-time libraries. On many systems, the various compilers are compatible and may be used interchangable. There are exceptions; if you use the `MPICH_F77` environment variable or the `-f77=name` command-line argument to override the choice of compiler and encounter problems, try reconfiguring MPICH2 with the new compiler, installing MPICH2 in a separate location. See the installation manual for more details.

Examples

To compile a single file `foo.f`, use

```
mpif77 -c foo.f
```

To link the output and make an executable, use

```
mpif77 -o foo foo.o
```

Combining compilation and linking in a single command

```
mpif77 -o foo foo.f
```

is a convenient way to build simple programs.

See Also

mpicc, mpicxx, mpif90, mpiexec

Location

`mpif77.txt`

3 MPI routines

MPI_Abort **MPI_Abort**

MPI_Abort — abort

Synopsis

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

Input Parameters

comm communicator of tasks to abort
errorcode error code to return to invoking environment

Notes

Terminates all MPI processes associated with the communicator `comm`; in most systems (all to date), terminates *all* processes.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`abort.c`

MPI_Accumulate — accumulate

Synopsis

```
int MPI_Accumulate(void *origin_addr, int origin_count, MPI_Datatype
                   origin_datatype, int target_rank, MPI_Aint
                   target_disp, int target_count, MPI_Datatype
                   target_datatype, MPI_Op op, MPI_Win win)
```

Input Parameters

origin_addr initial address of buffer (choice)
origin_count number of entries in buffer (nonnegative integer)
origin_datatype
: datatype of each buffer entry (handle)

target_rank rank of target (nonnegative integer)
target_disp displacement from start of window to beginning of target buffer (nonnegative integer)
target_count number of entries in target buffer (nonnegative integer)
target_datatype
: datatype of each entry in target buffer (handle)

op predefined reduce operation (handle)
win window object (handle)

Notes

The basic components of both the origin and target datatype must be the same predefined datatype (e.g., all `MPI_INT` or all `MPI_DOUBLE_PRECISION`).

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`accumulate.c`

MPI_Add_error_class — add error class

Synopsis

```
int MPI_Add_error_class(int *errorclass)
```

Output Parameter

errorclass New error class

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`add_error_class.c`

MPI_Add_error_code — add error code

Synopsis

```
int MPI_Add_error_code(int errorclass, int *errorcode)
```

Input Parameter

errorclass Error class to add an error code.

Output Parameter

errorcode New error code for this error class.

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler

may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_OTHER
: Other error; use `MPI_Error_string` to get more information about this error code.

Location

`add_error_code.c`

MPI_Add_error_string **MPI_Add_error_string**

MPI_Add_error_string — add error string

Synopsis

```
int MPI_Add_error_string(int errorcode, char *string)
```

Input Parameters

errorcode error code or class (integer)
string text corresponding to errorcode (string)

Notes

The string must be no more than `MPI_MAX_ERROR_STRING` characters long. The length of the string is as defined in the calling language. The length of the string does not include the null terminator in C or C++.

According to the MPI-2 standard, it is erroneous to call `MPI_Add_error_string` for an error code or class with a value less than or equal to `MPI_ERR_LASTCODE`. Thus, you cannot replace the predefined error messages with this routine.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`add_error_string.c`

MPI_Allgather **MPI_Allgather**

MPI_Allgather — Gathers data from all tasks and distribute it to all

Synopsis

```
int MPI_Allgather(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount,
```

Input Parameters

sendbuf starting address of send buffer (choice)
sendcount number of elements in send buffer (integer)
sendtype data type of send buffer elements (handle)
recvcount number of elements received from any process (integer)
recvtype data type of receive buffer elements (handle)
comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice)

Notes

The MPI standard (1.0 and 1.1) says that
The jth block of data sent from each proess is received by every process and placed in the jth block of the buffer `recvbuf`.
This is misleading; a better description is
The block of data sent from the jth process is received by every process and placed in the jth block of the buffer `recvbuf`.
This text was suggested by Rajeev Thakur and has been adopted as a clarification.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
: Invalid buffer pointer. Usually a null buffer where one is not valid.

Location

`allgather.c`

MPI_Allgatherv — Gathers data from all tasks and deliver it to all

Synopsis

```
int MPI_Allgatherv(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int *recvcount
```

Input Parameters

sendbuf starting address of send buffer (choice)
sendcount number of elements in send buffer (integer)
sendtype data type of send buffer elements (handle)
recvcounts integer array (of length group size) containing the number of elements that are received from each process

displs integer array (of length group size). Entry `i` specifies the displacement (relative to recvbuf) at which to place the incoming data from process `i`
recvtype data type of receive buffer elements (handle)
comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice)

Notes

The MPI standard (1.0 and 1.1) says that
The jth block of data sent from each proess is received by every process and placed in the jth block of the buffer `recvbuf`.
This is misleading; a better description is
The block of data sent from the jth process is received by every process and placed in the jth block of the buffer `recvbuf`.
This text was suggested by Rajeev Thakur, and has been adopted as a clarification to the MPI standard.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_ERR_BUFFER
Invalid buffer pointer. Usually a null buffer where one is not valid.

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`allgatherv.c`

MPI_Alloc_mem **MPI_Alloc_mem**

MPI_Alloc_mem — allocate memory for message passing and RMA

Synopsis

```
int MPI_Alloc_mem(MPI_Aint size, MPI_Info info, void *baseptr)
```

Input Parameters

size size of memory segment in bytes (nonnegative integer)
info info argument (handle)

Output Parameter

baseptr pointer to beginning of memory segment allocated

Notes

Using this routine from Fortran requires that the Fortran compiler accept a common pointer extension. See Section 4.11 (Memory Allocation) in the MPI-2 standard for more information and examples.
Also note that while `baseptr` is a `void *` type, this is simply to allow easy use of any pointer object for this parameter. In fact, this argument is really a `void **` type, that is, a pointer to a pointer.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_INFO
Invalid Info

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_NO_MEM
Insufficient memory available for allocation by `MPI_Alloc_mem`

Location

`alloc_mem.c`

MPI_Allreduce **MPI_Allreduce**

MPI_Allreduce — Combines values from all processes and distributes the result back to all processes

Synopsis

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

Input Parameters

sendbuf starting address of send buffer (choice)
count number of elements in send buffer (integer)
datatype data type of elements of send buffer (handle)
op operation (handle)
comm communicator (handle)

Output Parameter

recvbuf starting address of receive buffer (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Notes on collective operations

The reduction functions (`MPI_Op`) do not return an error value. As a result, if the functions detect an error, all they can do is either call `MPI_Abort` or silently skip the problem. Thus, if you change the error handler from `MPI_ERRORS_ARE_FATAL` to something else, for example, `MPI_ERRORS_RETURN`, then no error may be indicated.
The reason for this is the performance problems in ensuring that all collective routines return the same error value.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_ERR_BUFFER
: Invalid buffer pointer. Usually a null buffer where one is not valid.

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_OP
: Invalid operation. MPI operations (objects of type `MPI_Op`) must either be one of the predefined operations (e.g., `MPI_SUM`) or created with `MPI_Op_create`.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`allreduce.c`

MPI_Alltoall **MPI_Alltoall**

MPI_Alltoall — Sends data from all to all processes

Synopsis

```
int MPI_Alltoall(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, M
```

Input Parameters

sendbuf starting address of send buffer (choice)
sendcount number of elements to send to each process (integer)
sendtype data type of send buffer elements (handle)
recvcount number of elements received from any process (integer)
recvtype data type of receive buffer elements (handle)
comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
: Invalid buffer pointer. Usually a null buffer where one is not valid.

Location

`alltoall.c`

MPI_Alltoallv — Sends data from all to all processes, with a displacement

Synopsis

```
int MPI_Alltoallv(void *sendbuf, int *sendcnts, int *sdispls, MPI_Datatype sendtype, void *recvbuf, i
```

Input Parameters

sendbuf starting address of send buffer (choice)

sendcounts integer array equal to the group size specifying the number of elements to send to each processor

sdispls integer array (of length group size). Entry `j` specifies the displacement (relative to sendbuf from which to take the outgoing data destined for process `j`

sendtype data type of send buffer elements (handle)

recvcounts integer array equal to the group size specifying the maximum number of elements that can be received from each processor
rdispls integer array (of length group size). Entry `i` specifies the displacement (relative to recvbuf at which to place the incoming data from process `i`
recvtype data type of receive buffer elements (handle)
comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
Invalid buffer pointer. Usually a null buffer where one is not valid.

Location

`alltoallv.c`

MPI_Alltoallw **MPI_Alltoallw**

MPI_Alltoallw — Generalized all-to-all communication

Synopsis

```
int MPI_Alltoallw(void *sendbuf, int *sendcnts, int *sdispls, MPI_Datatype *sendtypes, void *recvbuf,
```

Input Parameters

sendbuf starting address of send buffer (choice)

sendcounts integer array equal to the group size specifying the number of elements to send to each processor (integer)

sdispls integer array (of length group size). Entry j specifies the displacement in bytes (relative to sendbuf) from which to take the outgoing data destined for process j

sendtypes array of datatypes (of length group size). Entry j specifies the type of data to send to process j (handle)

recvcounts integer array equal to the group size specifying the number of elements that can be received from each processor (integer)

rdispls integer array (of length group size). Entry i specifies the displacement in bytes (relative to recvbuf) at which to place the incoming data from process i

recvtypes array of datatypes (of length group size). Entry i specifies the type of data received from process i (handle)

comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`alltoallw.c`

MPI_Attr_delete **MPI_Attr_delete**

MPI_Attr_delete — Deletes attribute value associated with a key

Synopsis

```
int MPI_Attr_delete(MPI_Comm comm, int keyval)
```

Input Parameters

comm communicator to which attribute is attached (handle)
keyval The key value of the deleted attribute (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
This error class is associated with an error code that indicates that an attempt was made to free one of the permanent keys.

Location

attr_delete.c

MPI_Attr_get **MPI_Attr_get**

MPI_Attr_get — Retrieves attribute value by key

Synopsis

```
int MPI_Attr_get(MPI_Comm comm, int keyval, void *attr_value, int *flag)
```

Input Parameters

comm	communicator to which attribute is attached (handle)
keyval	key value (integer)

Output Parameters

attr_value	attribute value, unless `flag` = false
flag	true if an attribute value was extracted; false if no attribute is associated with the key

Notes

Attributes must be extracted from the same language as they were inserted in with `MPI_ATTR_PUT`. The notes for C and Fortran below explain why.

Notes for C

Even though the `attr_value` arguement is declared as `void *`, it is really the address of a void pointer. See the rationale in the standard for more details.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.
The `attr_value` in Fortran is a pointer to a Fortran integer, not a pointer to a `void *`.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler

MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in MPI_Comm_rank).

MPI_ERR_KEYVAL
Invalid keyval

Location

attr_get.c

MPI_Attr_put **MPI_Attr_put**

MPI_Attr_put — Stores attribute value associated with a key

Synopsis

```
int MPI_Attr_put(MPI_Comm comm, int keyval, void *attr_value)
```

Input Parameters

comm	communicator to which attribute will be attached (handle)
keyval	key value, as returned by MPI_KEYVAL_CREATE (integer)
attribute_val	attribute value

Notes

Values of the permanent attributes MPI_TAG_UB, MPI_HOST, MPI_IO, MPI_WTIME_IS_GLOBAL, MPI_UNIVERSE_SIZE, MPI_LASTUSEDCODE, and MPI_APPNUM may not be changed.
The type of the attribute value depends on whether C or Fortran is being used. In C, an attribute value is a pointer (void *); in Fortran, it is a single integer (*not* a pointer, since Fortran has no pointers and there are systems for which a pointer does not fit in an integer (e.g., any > 32 bit address system that uses 64 bits for Fortran DOUBLE PRECISION).
If an attribute is already present, the delete function (specified when the corresponding keyval was created) will be called.

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_KEYVAL
: Invalid keyval

MPI_ERR_ARG
: This error class is associated with an error code that indicates that an attempt was made to free one of the permanent keys.

See Also

MPI_Attr_get, MPI_Keyval_create, MPI_Attr_delete

Location

`attr_put.c`

MPI_Barrier **MPI_Barrier**

MPI_Barrier — Blocks until all process have reached this routine.

Synopsis

```
int MPI_Barrier( MPI_Comm comm )
```

Input Parameter

comm communicator (handle)

Notes

Blocks the caller until all group members have called it; the call returns at any process only after all group members have entered the call.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`barrier.c`

MPI_Bcast **MPI_Bcast**

MPI_Bcast — Broadcasts a message from the process with rank "root" to all other processes of the group.

Synopsis

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

Input/output Parameters

buffer starting address of buffer (choice)
count number of entries in buffer (integer)
datatype data type of buffer (handle)
root rank of broadcast root (integer)
comm communicator (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return

value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
: Invalid buffer pointer. Usually a null buffer where one is not valid.

MPI_ERR_ROOT
: Invalid root. The root must be specified as a rank in the communicator. Ranks must be between zero and the size of the communicator minus one.

Location

`bcast.c`

MPI_Bsend — Basic send with user-specified buffering

Synopsis

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

Input Parameters

buf initial address of send buffer (choice)
count number of elements in send buffer (nonnegative integer)
datatype datatype of each send buffer element (handle)

dest rank of destination (integer)
tag message tag (integer)
comm communicator (handle)

Notes

This send is provided as a convenience function; it allows the user to send messages without worring about where they are buffered (because the user *must* have provided buffer space with `MPI_Buffer_attach`).
In deciding how much buffer space to allocate, remember that the buffer space is not available for reuse by subsequent `MPI_Bsend`s unless you are certain that the message has been received (not just that it should have been received). For example, this code does not allocate enough buffer space

```
MPI_Buffer_attach( b, n*sizeof(double) + MPI_BSEND_OVERHEAD );
for (i=0; i<m; i++) {
    MPI_Bsend( buf, n, MPI_DOUBLE, ... );
}
```

because only enough buffer space is provided for a single send, and the loop may start a second `MPI_Bsend` before the first is done making use of the buffer.
In C, you can force the messages to be delivered by

```
MPI_Buffer_detach( &b, &n );
MPI_Buffer_attach( b, n );
```

(The `MPI_Buffer_detach` will not complete until all buffered messages are delivered.)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

See Also

MPI_Buffer_attach, MPI_Ibsend, MPI_Bsend_init

Location

`bsend.c`

MPI_Bsend_init **MPI_Bsend_init**

MPI_Bsend_init — Builds a handle for a buffered send

Synopsis

```
int MPI_Bsend_init(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI
```

Input Parameters

buf	initial address of send buffer (choice)
count	number of elements sent (integer)
datatype	type of each element (handle)
dest	rank of destination (integer)
tag	message tag (integer)
comm	communicator (handle)

Output Parameter

request	communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_TAG
: Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

Location

`bsend_init.c`

MPI_Buffer_attach **MPI_Buffer_attach**

MPI_Buffer_attach — Attaches a user-defined buffer for sending

Synopsis

```
int MPI_Buffer_attach(void *buffer, int size)
```

Input Parameters

buffer initial buffer address (choice)
size buffer size, in bytes (integer)

Notes

The size given should be the sum of the sizes of all outstanding Bsends that you intend to have, plus `MPI_BSEND_OVERHEAD` for each Bsend that you do. For the purposes of calculating size, you should use `MPI_Pack_size`. In other words, in the code

```
MPI_Buffer_attach( buffer, size );
MPI_Bsend( ..., count=20, datatype=type1,  ... );
...
MPI_Bsend( ..., count=40, datatype=type2, ... );
```

the value of `size` in the `MPI_Buffer_attach` call should be greater than the value computed by

```
MPI_Pack_size( 20, type1, comm, &s1 );
MPI_Pack_size( 40, type2, comm, &s2 );
size = s1 + s2 + 2 * MPI_BSEND_OVERHEAD;
```

The `MPI_BSEND_OVERHEAD` gives the maximum amount of space that may be used in the buffer for use by the BSEND routines in using the buffer. This value is in `mpi.h` (for C) and `mpif.h` (for Fortran).

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_BUFFER
: Invalid buffer pointer. Usually a null buffer where one is not valid.

MPI_ERR_INTERN
: An internal error has been detected. This is fatal. Please send a bug report to `mpi-bugs@mcs.anl.gov`.

See Also

MPI_Buffer_detach, MPI_Bsend

Location

bufattach.c

MPI_Buffer_detach **MPI_Buffer_detach**

MPI_Buffer_detach — Removes an existing buffer (for use in MPI_Bsend etc)

Synopsis

```
int MPI_Buffer_detach(void *buffer, int *size)
```

Output Parameters

buffer	initial buffer address (choice)
size	buffer size, in bytes (integer)

Notes

The reason that MPI_Buffer_detach returns the address and size of the buffer being detached is to allow nested libraries to replace and restore the buffer. For example, consider

```
int size, mysize, idummy;
void *ptr, *myptr, *dummy;
MPI_Buffer_detach( &ptr, &size );
MPI_Buffer_attach( myptr, mysize );
...
... library code ...
...
MPI_Buffer_detach( &dummy, &idummy );
MPI_Buffer_attach( ptr, size );
```

This is much like the action of the Unix signal routine and has the same strengths (it is simple) and weaknesses (it only works for nested usages).
Note that for this approach to work, MPI_Buffer_detach must return MPI_SUCCESS even when there is no buffer to detach. In that case, it returns a size of zero. The MPI 1.1 standard for MPI_BUFFER_DETACH contains the text

```
The statements made in this section describe the behavior of MPI for
buffered-mode sends. When no buffer is currently associated, MPI behaves
as if a zero-sized buffer is associated with the process.
```

This could be read as applying only to the various Bsend routines. This implementation takes the position that this applies to MPI_BUFFER_DETACH as well.

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.
The Fortran binding for this routine is different. Because Fortran does not have pointers, it is impossible to provide a way to use the output of this routine to exchange buffers. In this case, only the size field is set.

Notes for C

Even though the `bufferptr` argument is declared as `void *`, it is really the address of a void pointer. See the rationale in the standard for more details.

Location

`buffree.c`

MPI_Cancel **MPI_Cancel**

MPI_Cancel — Cancels a communication request

Synopsis

```
int MPI_Cancel(MPI_Request *request)
```

Input Parameter

request communication request (handle)

Note

Cancel has only been implemented for receive requests; it is a no-op for send requests. The primary expected use of MPI_Cancel is in multi-buffering schemes, where speculative MPI_Irecvs are made. When the computation completes, some of these receive requests may remain; using MPI_Cancel allows the user to cancel these unsatisfied requests.
Cancelling a send operation is much more difficult, in large part because the send will usually be at least partially complete (the information on the tag, size, and source are usually sent immediately to the destination). As of version 1.2.0, MPICH supports cancelling of sends. Users are advised that cancelling a send, while a local operation (as defined by the MPI standard), is likely to be expensive (usually generating one or more internal messages).

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Null Handles

The MPI 1.1 specification, in the section on opaque objects, explicitly

disallows freeing a null communicator. The text from the standard is

```
A null handle argument is an erroneous IN argument in MPI calls, unless an
exception is explicitly stated in the text that defines the function. Such
exception is allowed for handles to request objects in Wait and Test calls
(sections Communication Completion and Multiple Completions ). Otherwise, a
null handle can only be passed to a function that allocates a new object and
returns a reference to it in the handle.
```

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_REQUEST
: Invalid `MPI_Request`. Either null or, in the case of a `MPI_Start` or `MPI_Startall`, not a persistent request.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cancel.c`

MPI_Cart_coords **MPI_Cart_coords**

MPI_Cart_coords — Determines process coords in cartesian topology given rank in group

Synopsis

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

Input Parameters

comm communicator with cartesian structure (handle)
rank rank of a process within group of `comm` (integer)
maxdims length of vector `coords` in the calling program (integer)

Output Parameter

coords integer array (of size `ndims`) containing the Cartesian coordinates of specified process (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_DIMS
Invalid dimension argument. A dimension argument is null or its length is less than or equal to zero.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cart_coords.c`

MPI_Cart_create **MPI_Cart_create**

MPI_Cart_create — Makes a new communicator to which topology information has been attached

Synopsis

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

Input Parameters

comm_old input communicator (handle)
ndims number of dimensions of cartesian grid (integer)
dims integer array of size ndims specifying the number of processes in each dimension
periods logical array of size ndims specifying whether the grid is periodic (true) or not (false) in each dimension
reorder ranking may be reordered (true) or not (false) (logical)

Output Parameter

comm_cart communicator with new cartesian topology (handle)

Algorithm

We ignore `reorder` info currently.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_DIMS
Invalid dimension argument. A dimension argument is null or its length is less than or equal to zero.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cart_create.c`

MPI_Cart_get **MPI_Cart_get**

MPI_Cart_get — Retrieves Cartesian topology information associated with a communicator

Synopsis

```
int MPI_Cart_get(MPI_Comm comm, int maxdims, int *dims, int *periods, int *coords)
```

Input Parameters

comm communicator with cartesian structure (handle)
maxdims length of vectors `dims`, `periods`, and `coords` in the calling program (integer)

Output Parameters

dims number of processes for each cartesian dimension (array of integer)
periods periodicity (true/false) for each cartesian dimension (array of logical)
coords coordinates of calling process in cartesian structure (array of integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cart_get.c`

MPI_Cart_map **MPI_Cart_map**

MPI_Cart_map — Maps process to Cartesian topology information

Synopsis

```
int MPI_Cart_map(MPI_Comm comm_old, int ndims, int *dims, int *periods,
                 int *newrank)
```

Input Parameters

comm input communicator (handle)
ndims number of dimensions of Cartesian structure (integer)
dims integer array of size `ndims` specifying the number of processes in each coordinate direction
periods logical array of size `ndims` specifying the periodicity specification in each coordinate direction

Output Parameter

newrank reordered rank of the calling process; `MPI_UNDEFINED` if calling process does not belong to grid (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS

No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_DIMS
Invalid dimension argument. A dimension argument is null or its length is less than or equal to zero.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cart_map.c`

MPI_Cart_rank **MPI_Cart_rank**

MPI_Cart_rank — Determines process rank in communicator given Cartesian location

Synopsis

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

Input Parameters

comm communicator with cartesian structure (handle)

coords integer array (of size `ndims`, the number of dimensions of the Cartesian topology associated with `comm`) specifying the cartesian coordinates of a process

Output Parameter

rank rank of specified process (integer)

Notes

Out-of-range coordinates are erroneous for non-periodic dimensions. Versions of MPICH before 1.2.2 returned `MPI_PROC_NULL` for the rank in this case.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
: Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cart_rank.c`

MPI_Cart_shift — Returns the shifted source and destination ranks, given a shift direction and amount

Synopsis

```
int MPI_Cart_shift(MPI_Comm comm, int direction, int displ, int *source,
                   int *dest)
```

Input Parameters

comm communicator with cartesian structure (handle)
direction coordinate dimension of shift (integer)
displ displacement (> 0: upwards shift, < 0: downwards shift) (integer)

Output Parameters

source rank of source process (integer)
dest rank of destination process (integer)

Notes

The `direction` argument is in the range `[0,n-1]` for an n-dimensional Cartesian mesh.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
: Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cart_shift.c`

MPI_Cart_sub **MPI_Cart_sub**

MPI_Cart_sub — Partitions a communicator into subgroups which form lower-dimensional cartesian subgrids

Synopsis

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

Input Parameters

comm communicator with cartesian structure (handle)
remain_dims the ith entry of remain_dims specifies whether the ith dimension is kept in the subgrid (true) or is dropped (false) (logical vector)

Output Parameter

newcomm communicator containing the subgrid that includes the calling process (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cart_sub.c`

MPI_Cartdim_get **MPI_Cartdim_get**

MPI_Cartdim_get — Retrieves Cartesian topology information associated with a communicator

Synopsis

```
int MPI_Cartdim_get(MPI_Comm comm, int *ndims)
```

Input Parameter

comm communicator with cartesian structure (handle)

Output Parameter

ndims number of dimensions of the cartesian structure (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`cartdim_get.c`

MPI_Close_port **MPI_Close_port**

MPI_Close_port — close port

Synopsis

```
int MPI_Close_port(char *port_name)
```

Input Parameter

port_name a port name (string)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`close_port.c`

MPI_Comm_accept — Accept a request to form a new intercommunicator

Synopsis

```
int MPI_Comm_accept(char *port_name, MPI_Info info, int root, MPI_Comm comm, MPI_Comm *newcomm)
```

Input Parameters

port_name port name (string, used only on root)
info implementation-dependent information (handle, used only on root)
root rank in comm of root node (integer)
IN comm intracommunicator over which call is collective (handle)

Output Parameter

newcomm intercommunicator with client as remote group (handle)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_INFO
: Invalid Info

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`comm_accept.c`

MPI_Comm_call_errhandler **MPI_Comm_call_errhandler**

MPI_Comm_call_errhandler — Call the error handler installed on a communicator

Synopsis

```
int MPI_Comm_call_errhandler(MPI_Comm comm, int errorcode)
```

Input Parameters

comm communicator with error handler (handle)
errorcode error code (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`comm_call_errhandler.c`

MPI_Comm_compare **MPI_Comm_compare**

MPI_Comm_compare — Compares two communicators

Synopsis

```
int MPI_Comm_compare(MPI_Comm comm1, MPI_Comm comm2, int *result)
```

Input Parameters

comm1 comm1 (handle)
comm2 comm2 (handle)

Output Parameter

result integer which is `MPI_IDENT` if the contexts and groups are the same, `MPI_CONGRUENT` if different contexts but identical groups, `MPI_SIMILAR` if different contexts but similar groups, and `MPI_UNEQUAL` otherwise

Using 'MPI_COMM_NULL' with 'MPI_Comm_compare'

It is an error to use `MPI_COMM_NULL` as one of the arguments to `MPI_Comm_compare`. The relevant sections of the MPI standard are
.(2.4.1 Opaque Objects)A null handle argument is an erroneous `IN` argument in MPI calls, unless an exception is explicitly stated in the text that defines the function.
.(5.4.1. Communicator Accessors)where there is no text in `MPI_COMM_COMPARE` allowing a null handle.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`comm_compare.c`

MPI_Comm_connect **MPI_Comm_connect**

MPI_Comm_connect — Make a request to form a new intercommunicator

Synopsis

```
int MPI_Comm_connect(char *port_name, MPI_Info info, int root, MPI_Comm comm, MPI_Comm *newcomm)
```

Input Parameters

port_name network address (string, used only on root)
info implementation-dependent information (handle, used only on root)
root rank in comm of root node (integer)

comm intracommunicator over which call is collective (handle)

Output Parameter

newcomm intercommunicator with server as remote group (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_INFO
Invalid Info

MPI_ERR_PORT

Location

`comm_connect.c`

MPI_Comm_create **MPI_Comm_create**

MPI_Comm_create — Creates a new communicator

Synopsis

```
int MPI_Comm_create(MPI_Comm comm, MPI_Group group, MPI_Comm *newcomm)
```

Input Parameters

comm communicator (handle)
group group, which is a subset of the group of `comm` (handle)

Output Parameter

comm_out new communicator (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_GROUP
Null or invalid group passed to function.

See Also

MPI_Comm_free

Location

`comm_create.c`

MPI_Comm_create_errhandler **MPI_Comm_create_errhandler**

MPI_Comm_create_errhandler — Create a communicator error handler

Synopsis

```
int MPI_Comm_create_errhandler(MPI_Comm_errhandler_fn *function, MPI_Errhandler *errhandler)
```

Input Parameter

function user defined error handling procedure (function)

Output Parameter

errhandler MPI error handler (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_OTHER
: Other error; use `MPI_Error_string` to get more information about this error code.

Location

`comm_create_errhandler.c`

MPI_Comm_create_keyval — Create a new attribute key

Synopsis

```
int MPI_Comm_create_keyval(MPI_Comm_copy_attr_function *comm_copy_attr_fn,
                           MPI_Comm_delete_attr_function *comm_delete_attr_fn,
                           int *comm_keyval, void *extra_state)
```

Input Parameters

MPI_Comm_copy_attr_function *comm_copy_attr_fn
: copy function

MPI_Comm_delete_attr_function *comm_delete_attr_fn
: delete function

void *extra_state
: extra state

Output Parameters

int *comm_keyval
: keyval

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`comm_create_keyval.c`

MPI_Comm_delete_attr **MPI_Comm_delete_attr**

MPI_Comm_delete_attr — delete communicator attribute

Synopsis

```
int MPI_Comm_delete_attr(MPI_Comm comm, int comm_keyval)
```

Input Parameters

comm communicator to which attribute is attached (handle)
comm_keyval The key value of the deleted attribute (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
This error class is associated with an error code that indicates that an attempt was made to free one of the permanent keys.

Location

`comm_delete_attr.c`

MPI_Comm_disconnect **MPI_Comm_disconnect**

MPI_Comm_disconnect — Disconnect from a communicator

Synopsis

```
int MPI_Comm_disconnect(MPI_Comm *comm)
```

Input Parameter
comm communicator (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`comm_disconnect.c`

MPI_Comm_dup **MPI_Comm_dup**

MPI_Comm_dup — Duplicates an existing communicator with all its cached information

Synopsis

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

Input Parameter

comm communicator (handle)

Output Parameter

newcomm A new communicator over the same group as `comm` but with a new context. See notes. (handle)

Notes

This routine is used to create a new communicator that has a new communication context but contains the same group of processes as the input communicator. Since all MPI communication is performed within a communicator (specifies as the group of processes *plus* the context), this routine provides an effective way to create a private communicator for use by a software module or library. In particular, no library routine should use `MPI_COMM_WORLD` as the communicator; instead, a duplicate of a user-specified communicator should always be used. For more information, see Using MPI, 2nd edition.
Because this routine essentially produces a copy of a communicator, it also copies any attributes that have been defined on the input communicator, using the attribute copy function specified by the `copy_function` argument to `MPI_Keyval_create`. This is particularly useful for (a) attributes that describe some property of the group associated with the communicator, such as its interconnection topology and (b) communicators that are given back to the user; the attibutes in this case can track subsequent `MPI_Comm_dup` operations on this communicator.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

See Also

MPI_Comm_free, MPI_Keyval_create, MPI_Attr_put, MPI_Attr_delete, MPI_Comm_create_keyval, MPI_Comm_set_attr, MPI_Comm_delete_attr

Location

`comm_dup.c`

MPI_Comm_free **MPI_Comm_free**

MPI_Comm_free — Marks the communicator object for deallocation

Synopsis

```
int MPI_Comm_free(MPI_Comm *comm)
```

Input Parameter

comm communicator to be destroyed (handle)

Notes

This routine *frees* a communicator. Because the communicator may still be in use by other MPI routines, the actual communicator storage will not be freed until all references to this communicator are removed. For most users, the effect of this routine is the same as if it was in fact freed at this time of this call.

Null Handles

The MPI 1.1 specification, in the section on opaque objects, explicitly

disallows freeing a null communicator. The text from the standard is

```
A null handle argument is an erroneous IN argument in MPI calls, unless an
exception is explicitly stated in the text that defines the function. Such
exception is allowed for handles to request objects in Wait and Test calls
(sections Communication Completion and Multiple Completions ). Otherwise, a
null handle can only be passed to a function that allocates a new object and
returns a reference to it in the handle.
```

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler

MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in MPI_Comm_rank).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., MPI_ERR_RANK).

Location

comm_free.c

MPI_Comm_free_keyval **MPI_Comm_free_keyval**

MPI_Comm_free_keyval — free communicator keyval

Synopsis

```
int MPI_Comm_free_keyval(int *comm_keyval)
```

Input Parameter

keyval Frees the integer key value (integer)

Notes

Key values are global (they can be used with any and all communicators)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not*

guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_ARG
This error class is associated with an error code that indicates that an attempt was made to free one of the permanent keys.

Location

`comm_free_keyval.c`

MPI_Comm_get_attr **MPI_Comm_get_attr**

MPI_Comm_get_attr — get communicator attribute

Synopsis

```
int MPI_Comm_get_attr(MPI_Comm comm, int comm_keyval, void *attribute_val, int *flag)
```

Input Parameters

comm communicator to which attribute is attached (handle)
keyval key value (integer)

Output Parameters

attr_value attribute value, unless `flag` = false
flag true if an attribute value was extracted; false if no attribute is associated with the key

Notes

Attributes must be extracted from the same language as they were inserted in with `MPI_Comm_set_attr`. The notes for C and Fortran below explain why.

Notes for C

Even though the `attr_value` arguement is declared as `void *`, it is really the address of a void pointer. See the rationale in the standard for more details.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return

value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_KEYVAL
: Invalid keyval

Location

`comm_get_attr.c`

MPI_Comm_get_errhandler **MPI_Comm_get_errhandler**

MPI_Comm_get_errhandler — Get the error handler attached to a communicator

Synopsis

```
int MPI_Comm_get_errhandler(MPI_Comm comm, MPI_Errhandler *errhandler)
```

Input Parameter

comm communicator (handle)

Output Parameter

errhandler perror handler currently associated with communicator (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`comm_get_errhandler.c`

MPI_Comm_get_name **MPI_Comm_get_name**

MPI_Comm_get_name — return the print name from the communicator

Synopsis

```
int MPI_Comm_get_name(MPI_Comm comm, char *comm_name, int *resultlen)
```

Input Parameter

comm Communicator to get name of (handle)

Output Parameters

comm_name One output, contains the name of the communicator. It must be an array of size at least `MPI_MAX_NAME_STRING`.
resultlen Number of characters in name

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`comm_get_name.c`

MPI_Comm_get_parent **MPI_Comm_get_parent**

MPI_Comm_get_parent — short description

Synopsis

```
int MPI_Comm_get_parent(MPI_Comm *parent)
```

Output Parameter

parent the parent communicator (handle)

Notes

If a process was started with `MPI_Comm_spawn` or `MPI_Comm_spawn_multiple`, `MPI_Comm_get_parent` returns the parent intercommunicator of the current process. This parent intercommunicator is created implicitly inside of `MPI_Init` and is the same intercommunicator returned by `MPI_Comm_spawn` in the parents.
If the process was not spawned, `MPI_Comm_get_parent` returns `MPI_COMM_NULL`.
After the parent communicator is freed or disconnected, `MPI_Comm_get_parent` returns `MPI_COMM_NULL`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`comm_get_parent.c`

MPI_Comm_group **MPI_Comm_group**

MPI_Comm_group — Accesses the group associated with given communicator

Synopsis

```
int MPI_Comm_group(MPI_Comm comm, MPI_Group *group)
```

Input Parameter

comm Communicator

Output Parameter

group Group in communicator

Using 'MPI_COMM_NULL' with 'MPI_Comm_group'

It is an error to use `MPI_COMM_NULL` as one of the arguments to `MPI_Comm_group`. The relevant sections of the MPI standard are

.(2.4.1 Opaque Objects)A null handle argument is an erroneous `IN` argument in MPI calls, unless an exception is explicitly stated in the text that defines the function.
.(5.3.2. Group Constructors)<no text in `MPI_COMM_GROUP` allowing a null handle>
Previous versions of MPICH allow `MPI_COMM_NULL` in this function. In the interests of promoting portability of applications, we have changed the behavior of `MPI_Comm_group` to detect this violation of the MPI standard.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`comm_group.c`

MPI_Comm_join **MPI_Comm_join**

MPI_Comm_join — join

Synopsis

```
int MPI_Comm_join(int fd, MPI_Comm *intercomm)
```

Input Parameter

fd socket file descriptor

Output Parameter

intercomm new intercommunicator (handle)

Notes

The socket must be quiescent before `MPI_COMM_JOIN` is called and after `MPI_COMM_JOIN` returns. More specifically, on entry to `MPI_COMM_JOIN`, a read on the socket will not read any data that was written to the socket before the remote process called `MPI_COMM_JOIN`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`comm_join.c`

MPI_Comm_rank **MPI_Comm_rank**

MPI_Comm_rank — Determines the rank of the calling process in the communicator

Synopsis

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

Input Argument

comm communicator (handle)

Output Argument

rank rank of the calling process in group of `comm` (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`comm_rank.c`

MPI_Comm_remote_group **MPI_Comm_remote_group**

MPI_Comm_remote_group — Accesses the remote group associated with the given inter-communicator

Synopsis

```
int MPI_Comm_remote_group(MPI_Comm comm, MPI_Group *group)
```

Input Parameter

comm Communicator (must be intercommunicator)

Output Parameter

group remote group of communicator

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

`comm_remote_group.c`

MPI_Comm_remote_size — Determines the size of the remote group associated with an inter-communictor

Synopsis

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

Input Parameter

comm communicator (handle)

Output Parameter

size number of processes in the group of `comm` (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`comm_remote_size.c`

MPI_Comm_set_attr **MPI_Comm_set_attr**

MPI_Comm_set_attr — set communicator attribute

Synopsis

```
int MPI_Comm_set_attr(MPI_Comm comm, int comm_keyval, void *attribute_val)
```

Input Parameters

comm communicator to which attribute will be attached (handle)
keyval key value, as returned by `MPI_Comm_create_keyval` (integer)
attribute_val attribute value

Notes

Values of the permanent attributes `MPI_TAG_UB`, `MPI_HOST`, `MPI_IO`, `MPI_WTIME_IS_GLOBAL`, `MPI_UNIVERSE_SIZE`, `MPI_LASTUSEDCODE`, and `MPI_APPNUM` may not be changed.

The type of the attribute value depends on whether C or Fortran is being used. In C, an attribute value is a pointer (`void *`); in Fortran, it is an address-sized integer.
If an attribute is already present, the delete function (specified when the corresponding keyval was created) will be called.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_KEYVAL
: Invalid keyval

MPI_ERR_ARG
: This error class is associated with an error code that indicates that an attempt was made to free one of the permanent keys.

Location

`comm_set_attr.c`

MPI_Comm_set_errhandler **MPI_Comm_set_errhandler**

MPI_Comm_set_errhandler — Set the error handler for a communicator

Synopsis

```
int MPI_Comm_set_errhandler(MPI_Comm comm, MPI_Errhandler errhandler)
```

Input Parameters

comm communicator (handle)
errhandler new error handler for communicator (handle)

Arguments

MPI_Comm comm
communicator

MPI_Errhandler errhandler
error handler

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`comm_set_errhandler.c`

MPI_Comm_set_name **MPI_Comm_set_name**

MPI_Comm_set_name — set the communicator name

Synopsis

```
int MPI_Comm_set_name(MPI_Comm comm, char *comm_name)
```

Input Parameters

MPI_Comm comm
: communicator to name (handle)

char *comm_name
: Name for communicator

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

Location

```
comm_set_name.c
```

MPI_Comm_size **MPI_Comm_size**

MPI_Comm_size — Determines the size of the group associated with a communictor

Synopsis

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

Input Argument

comm communicator (handle)

Output Argument

size number of processes in the group of `comm` (integer)

Notes

`MPI_COMM_NULL` is *not* a valid argument to this function.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`comm_size.c`

MPI_Comm_spawn **MPI_Comm_spawn**

MPI_Comm_spawn — spawn up to maxprocs instances of a single mpi application

Synopsis

```
int MPI_Comm_spawn(char *command, char *argv[], int maxprocs, MPI_Info info,
                   int root, MPI_Comm comm, MPI_Comm *intercomm,
                   int array_of_errcodes[])
```

Input Parameters

command name of program to be spawned (string, significant only at root)
argv arguments to command (array of strings, significant only at root)
maxprocs maximum number of processes to start (integer, significant only at root)
info a set of key-value pairs telling the runtime system where and how to start the processes (handle, significant only at root)
root rank of process in which previous arguments are examined (integer)
comm intracommunicator containing group of spawning processes (handle)

Output Parameters

intercomm intercommunicator between original group and the newly spawned group (handle)
array_of_errcodes
one code per process (array of integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM

Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_INFO
Invalid Info

Location

`comm_spawn.c`

MPI_Comm_split **MPI_Comm_split**

MPI_Comm_split — Creates new communicators based on colors and keys

Synopsis

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

Input Parameters

comm communicator (handle)
color control of subset assignment (nonnegative integer). Processes with the same color are in the same new communicator
key control of rank assigment (integer)

Output Parameter

newcomm new communicator (handle)

Notes

The `color` must be non-negative or `MPI_UNDEFINED`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Algorithm

1. Use MPI_Allgather to get the color and key from each process
2. Count the number of processes with the same color; create a
   communicator with that many processes. If this process has
   {\tt MPI_UNDEFINED} as the color, create a process with a single member.

```
3. Use key to order the ranks
4. Set the VCRs using the ordered key values
```

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

See Also

MPI_Comm_free

Location

`comm_split.c`

MPI_Comm_test_inter **MPI_Comm_test_inter**

MPI_Comm_test_inter — Tests to see if a comm is an inter-communicator

Synopsis

```
int MPI_Comm_test_inter(MPI_Comm comm, int *flag)
```

Input Parameter

comm communicator (handle)

Output Parameter

flag (logical)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`comm_test_inter.c`

MPI_Dims_create **MPI_Dims_create**

MPI_Dims_create — Creates a division of processors in a cartesian grid

Synopsis

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

Input Parameters

nnodes number of nodes in a grid (integer)
ndims number of cartesian dimensions (integer)

In/Out Parameter

dims integer array of size `ndims` specifying the number of nodes in each dimension. A value of 0 indicates that `MPI_Dims_create` should fill in a suitable value.

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`dims_create.c`

MPI_Errhandler_create **MPI_Errhandler_create**

MPI_Errhandler_create — Creates an MPI-style errorhandler

Synopsis

```
int MPI_Errhandler_create(MPI_Handler_function *function, MPI_Errhandler *errhandler)
```

Input Parameter

function user defined error handling procedure

Output Parameter

errhandler MPI error handler (handle)

Notes

The MPI Standard states that an implementation may make the output value (errhandler) simply the address of the function. However, the action of `MPI_Errhandler_free` makes this impossible, since it is required to set the value of the argument to `MPI_ERRHANDLER_NULL`. In addition, the actual error handler must remain until all communicators that use it are freed.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

`errhandler_create.c`

MPI_Errhandler_free **MPI_Errhandler_free**

MPI_Errhandler_free — Frees an MPI-style errorhandler

Synopsis

```
int MPI_Errhandler_free(MPI_Errhandler *errhandler)
```

Input Parameter

errhandler MPI error handler (handle). Set to `MPI_ERRHANDLER_NULL` on exit.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`errhandler_free.c`

MPI_Errhandler_get — Gets the error handler for a communicator

Synopsis

```
int MPI_Errhandler_get(MPI_Comm comm, MPI_Errhandler *errhandler)
```

Input Parameter

comm communicator to get the error handler from (handle)

Output Parameter

errhandler MPI error handler currently associated with communicator (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Note on Implementation

The MPI Standard was unclear on whether this routine required the user to call `MPI_Errhandler_free` once for each call made to this routine in order to free the error handler. After some debate, the MPI Forum added an explicit statement that users are required to call

MPI_Errhandler_free when the return value from this routine is no longer needed. This behavior is similar to the other MPI routines for getting objects; for example, MPI_Comm_group requires that the user call MPI_Group_free when the group returned by MPI_Comm_group is no longer needed.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in MPI_Comm_rank).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., MPI_ERR_RANK).

Location

errhandler_get.c

MPI_Errhandler_set **MPI_Errhandler_set**

MPI_Errhandler_set — Sets the error handler for a communicator

Synopsis

```
int MPI_Errhandler_set(MPI_Comm comm, MPI_Errhandler errhandler)
```

Input Parameters

comm communicator to set the error handler for (handle)
errhandler new MPI error handler for communicator (handle)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`errhandler_set.c`

MPI_Error_class **MPI_Error_class**

MPI_Error_class — Converts an error code into an error class

Synopsis

```
int MPI_Error_class(int errorcode, int *errorclass)
```

Input Parameter

errorcode Error code returned by an MPI routine

Output Parameter

errorclass Error class associated with `errorcode`

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`error_class.c`

MPI_Error_string — Return a string for a given error code

Synopsis

```
int MPI_Error_string(int errorcode, char *string, int *resultlen)
```

Input Parameters

errorcode Error code returned by an MPI routine or an MPI error class

Output Parameter

string Text that corresponds to the errorcode
resultlen Length of string

Notes: Error codes are the values return by MPI routines (in C) or in the `ierr` argument (in Fortran). These can be converted into error classes with the routine `MPI_Error_class`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler

may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`error_string.c`

MPI_Exscan **MPI_Exscan**

MPI_Exscan — Computes the exclusive scan (partial reductions) of data on a collection of processes

Synopsis

```
int MPI_Exscan(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm co
```

Input Parameters

sendbuf starting address of send buffer (choice)
count number of elements in input buffer (integer)
datatype data type of elements of input buffer (handle)
op operation (handle)
comm communicator (handle)

Output Parameter

recvbuf starting address of receive buffer (choice)

Notes

`MPI_Exscan` is like `MPI_Scan`, except that the contribution from the calling process is not included in the result at the calling process (it is contributed to the subsequent processes, of course).

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Notes on collective operations

The reduction functions (`MPI_Op`) do not return an error value. As a result, if the functions detect an error, all they can do is either call `MPI_Abort` or silently skip the problem. Thus, if you change the error handler from `MPI_ERRORS_ARE_FATAL` to something else, for example, `MPI_ERRORS_RETURN`, then no error may be indicated.
The reason for this is the performance problems in ensuring that all collective routines return the same error value.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
: Invalid buffer pointer. Usually a null buffer where one is not valid.

MPI_ERR_BUFFER
: This error class is associcated with an error code that indicates that two buffer arguments are *aliased*; that is, the describe overlapping storage (often the exact same storage). This is prohibited in MPI (because it is prohibited by the Fortran standard, and rather than have a separate case for C and Fortran, the MPI Forum adopted the more restrictive requirements of Fortran).

Location

`exscan.c`

MPI_File_call_errhandler **MPI_File_call_errhandler**

MPI_File_call_errhandler — Call the error handler installed on a file

Synopsis

```
int MPI_File_call_errhandler(MPI_File fh, int errorcode)
```

Input Parameters

fh file with error handler (handle)
errorcode error code (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_FILE
Invalid MPI File handle

Location

`file_call_errhandler.c`

MPI_File_create_errhandler — create file error handler

Synopsis

```
int MPI_File_create_errhandler(MPI_File_errhandler_fn *function, MPI_Errhandler *errhandler)
```

Arguments

MPI_File_errhandler_fn *function
function

MPI_Errhandler *errhandler
error handler

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`file_create_errhandler.c`

MPI_File_get_errhandler **MPI_File_get_errhandler**

MPI_File_get_errhandler — get file error handler

Synopsis

```
int MPI_File_get_errhandler(MPI_File file, MPI_Errhandler *errhandler)
```

Arguments

MPI_File file file

MPI_Errhandler *errhandler
: error handler

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return

value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`file_get_errhandler.c`

MPI_File_set_errhandler **MPI_File_set_errhandler**

MPI_File_set_errhandler — set file error handler

Synopsis

```
int MPI_File_set_errhandler(MPI_File file, MPI_Errhandler errhandler)
```

Arguments

MPI_File file file

MPI_Errhandler errhandler
: error handler

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`file_set_errhandler.c`

MPI_Finalize **MPI_Finalize**

MPI_Finalize — Terminates MPI execution environment

Synopsis

```
int MPI_Finalize( void )
```

Notes

All processes must call this routine before exiting. The number of processes running *after* this routine is called is undefined; it is best not to perform much more than a `return rc` after calling `MPI_Finalize`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not*

guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`finalize.c`

MPI_Finalized **MPI_Finalized**

MPI_Finalized — Indicates whether `MPI_Finalize` has been called.

Synopsis

```
int MPI_Finalized( int * flag )
```

Output Argument

flag Flag is true if `MPI_Finalize` has been called and false otherwise.

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`finalized.c`

MPI_Free_mem **MPI_Free_mem**

MPI_Free_mem — Free memory allocatd with MPI_Alloc_mem

Synopsis

```
int MPI_Free_mem(void *base)
```

Input Parameter

base initial address of memory segment allocated by `MPI_ALLOC_MEM` (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`free_mem.c`

MPI_Gather **MPI_Gather**

MPI_Gather — Gathers together values from a group of processes

Synopsis

```
int MPI_Gather(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvcnt, MPI_Dat
```

Input Parameters

sendbuf starting address of send buffer (choice)
sendcount number of elements in send buffer (integer)
sendtype data type of send buffer elements (handle)
recvcount number of elements for any single receive (integer, significant only at root)
recvtype data type of recv buffer elements (significant only at root) (handle)
root rank of receiving process (integer)
comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice, significant only at `root`)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
Invalid buffer pointer. Usually a null buffer where one is not valid.

Location

gather.c

MPI_Gatherv **MPI_Gatherv**

MPI_Gatherv — Gathers into specified locations from all processes in a group

Synopsis

```
int MPI_Gatherv(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int *recvcnts, int
```

Input Parameters

sendbuf starting address of send buffer (choice)
sendcount number of elements in send buffer (integer)
sendtype data type of send buffer elements (handle)
recvcounts integer array (of length group size) containing the number of elements that are received from each process (significant only at `root`)
displs integer array (of length group size). Entry `i` specifies the displacement relative to recvbuf at which to place the incoming data from process `i` (significant only at root)
recvtype data type of recv buffer elements (significant only at `root`) (handle)
root rank of receiving process (integer)
comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice, significant only at `root`)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS

No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
Invalid buffer pointer. Usually a null buffer where one is not valid.

Location

`gatherv.c`

MPI_Get **MPI_Get**

MPI_Get — Get data from a remote process

Synopsis

```
int MPI_Get(void *origin_addr, int origin_count, MPI_Datatype
            origin_datatype, int target_rank, MPI_Aint target_disp,
            int target_count, MPI_Datatype target_datatype, MPI_Win
            win)
```

Input Parameters

origin_addr Address of the buffer in which to receive the data
origin_count number of entries in origin buffer (nonnegative integer)
origin_datatype
datatype of each entry in origin buffer (handle)

target_rank rank of target (nonnegative integer)
target_disp displacement from window start to the beginning of the target buffer (nonnegative integer)
target_count number of entries in target buffer (nonnegative integer)
target_datatype
datatype of each entry in target buffer (handle)

win window object used for communication (handle)

Output Parameter

origin_addr initial address of origin buffer (choice)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_WIN
: Invalid MPI window object

Location

`get.c`

MPI_Get_address **MPI_Get_address**

MPI_Get_address — get address

Synopsis

```
int MPI_Get_address(void *location, MPI_Aint *address)
```

Input Argument

location location in caller memory (choice)

Output Argument

address address of location (address) Arguments:

Notes

This routine is provided for both the Fortran and C programmers. On many systems, the address returned by this routine will be the same as produced by the C `&` operator, but this is not required in C and may not be true of systems with word- rather than byte-oriented instructions or systems with segmented address spaces.
This routine should be used instead of `MPI_Address`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_OTHER
: Other error; use `MPI_Error_string` to get more information about this error code.

Location

`get_address.c`

MPI_Get_processor_name **MPI_Get_processor_name**

MPI_Get_processor_name — Gets the name of the processor

Synopsis

```
int MPI_Get_processor_name( char *name, int *resultlen)
```

Output Parameters

name A unique specifier for the actual (as opposed to virtual) node. This must be an array of size at least `MPI_MAX_PROCESSOR_NAME`.
resultlen Length (in characters) of the name

Notes

The name returned should identify a particular piece of hardware; the exact format is implementation defined. This name may or may not be the same as might be returned by `gethostname`, `uname`, or `sysinfo`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`getpname.c`

MPI_Get_version **MPI_Get_version**

MPI_Get_version — Return the version number of MPI

Synopsis

```
int MPI_Get_version( int *version, int *subversion )
```

Output Parameters

version Version of MPI
subversion Subversion of MPI

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`version.c`

MPI_Graph_create — Makes a new communicator to which topology information has been attached

Synopsis

```
int MPI_Graph_create(MPI_Comm comm_old, int nnodes, int *index, int *edges,
                     int reorder, MPI_Comm *comm_graph)
```

Input Parameters

comm_old input communicator without topology (handle)
nnodes number of nodes in graph (integer)
index array of integers describing node degrees (see below)
edges array of integers describing graph edges (see below)
reorder ranking may be reordered (true) or not (false) (logical)

Output Parameter

comm_graph communicator with graph topology added (handle)

Notes

Each process must provide a description of the entire graph, not just the neigbors of the calling process.

Algorithm

We ignore the `reorder` info currently.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
: Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`graphcreate.c`

MPI_Graph_get **MPI_Graph_get**

MPI_Graph_get — Retrieves graph topology information associated with a communicator

Synopsis

```
int MPI_Graph_get(MPI_Comm comm, int maxindex, int maxedges, int *index, int *edges)
```

Input Parameters

comm communicator with graph structure (handle)
maxindex length of vector `index` in the calling program (integer)
maxedges length of vector `edges` in the calling program (integer)

Output Parameter

index array of integers containing the graph structure (for details see the definition of `MPI_GRAPH_CREATE`)
edges array of integers containing the graph structure

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`graph_get.c`

MPI_Graph_map **MPI_Graph_map**

MPI_Graph_map — Maps process to graph topology information

Synopsis

```
int MPI_Graph_map(MPI_Comm comm_old, int nnodes, int *index, int *edges, int *newrank)
```

Input Parameters

comm input communicator (handle)
nnodes number of graph nodes (integer)
index integer array specifying the graph structure, see `MPI_GRAPH_CREATE`
edges integer array specifying the graph structure

Output Parameter

newrank reordered rank of the calling process; `MPI_UNDEFINED` if the calling process does not belong to graph (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`graph_map.c`

MPI_Graph_neighbors **MPI_Graph_neighbors**

MPI_Graph_neighbors — Returns the neighbors of a node associated with a graph topology

Synopsis

```
int MPI_Graph_neighbors(MPI_Comm comm, int rank, int maxneighbors,
                        int *neighbors)
```

Input Parameters

comm communicator with graph topology (handle)
rank rank of process in group of comm (integer)
maxneighbors
size of array neighbors (integer)

Output Parameters

neighbors ranks of processes that are neighbors to specified process (array of integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG

Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., MPI_ERR_RANK).

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (MPI_Recv, MPI_Irecv, MPI_Sendrecv, etc.) may also be MPI_ANY_SOURCE.

Location

graph_nbr.c

MPI_Graph_neighbors_count **MPI_Graph_neighbors_count**

MPI_Graph_neighbors_count — Returns the number of neighbors of a node associated with a graph topology

Synopsis

```
int MPI_Graph_neighbors_count(MPI_Comm comm, int rank, int *nneighbors)
```

Input Parameters

comm communicator with graph topology (handle)
rank rank of process in group of comm (integer)

Output Parameter

nneighbors number of neighbors of specified process (integer)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS

No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

Location

`graphnbrcnt.c`

MPI_Graphdims_get **MPI_Graphdims_get**

MPI_Graphdims_get — Retrieves graph topology information associated with a communicator

Synopsis

```
int MPI_Graphdims_get(MPI_Comm comm, int *nnodes, int *nedges)
```

Input Parameters

comm communicator for group with graph structure (handle)

Output Parameter

nnodes number of nodes in graph (integer)
nedges number of edges in graph (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler

may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TOPOLOGY
: Invalid topology. Either there is no topology associated with this communicator, or it is not the correct type (e.g., `MPI_CART` when expecting `MPI_GRAPH`).

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`graphdimsget.c`

MPI_Grequest_complete **MPI_Grequest_complete**

MPI_Grequest_complete — Notify MPI that a user-defined request is complete

Synopsis

```
int MPI_Grequest_complete( MPI_Request request )
```

Input Parameter

request Generalized request to mark as complete

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler

may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`greq_complete.c`

MPI_Grequest_start **MPI_Grequest_start**

MPI_Grequest_start — Create and return a user-defined request

Synopsis

```
int MPI_Grequest_start( MPI_Grequest_query_function *query_fn,
                        MPI_Grequest_free_function *free_fn,
                        MPI_Grequest_cancel_function *cancel_fn,
                        void *extra_state, MPI_Request *request )
```

Input Parameters

query_fn callback function invoked when request status is queried (function)
free_fn callback function invoked when request is freed (function)
cancel_fn callback function invoked when request is cancelled (function)
extra_state Extra state passed to the above functions.

Output Parameter

request Generalized request (handle)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current

MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`greq_start.c`

MPI_Group_compare **MPI_Group_compare**

MPI_Group_compare — Compares two groups

Synopsis

```
int MPI_Group_compare(MPI_Group group1, MPI_Group group2, int *result)
```

Input Parameters

group1 group1 (handle)
group2 group2 (handle)

Output Parameter

result integer which is `MPI_IDENT` if the order and members of the two groups are the same, `MPI_SIMILAR` if only the members are the same, and `MPI_UNEQUAL` otherwise

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler`

(for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_GROUP
Null or invalid group passed to function.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`group_compare.c`

MPI_Group_difference **MPI_Group_difference**

MPI_Group_difference — Makes a group from the difference of two groups

Synopsis

```
int MPI_Group_difference(MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)
```

Input Parameters

group1 first group (handle)
group2 second group (handle)

Output Parameter

newgroup difference group (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler`

(for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_GROUP
Null or invalid group passed to function.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

See Also

MPI_Group_free

Location

`group_difference.c`

MPI_Group_excl **MPI_Group_excl**

MPI_Group_excl — Produces a group by reordering an existing group and taking only unlisted members

Synopsis

```
int MPI_Group_excl(MPI_Group group, int n, int *ranks, MPI_Group *newgroup)
```

Input Parameters

group group (handle)
n number of elements in array `ranks` (integer)
ranks array of integer ranks in `group` not to appear in `newgroup`

Output Parameter

newgroup new group derived from above, preserving the order defined by `group` (handle)

Note

The MPI standard requires that each of the ranks to excluded must be a valid rank in the group and all elements must be distinct or the function is erroneous.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_GROUP
: Null or invalid group passed to function.

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

See Also

MPI_Group_free

Location

`group_excl.c`

MPI_Group_free **MPI_Group_free**

MPI_Group_free — Frees a group

Synopsis

```
int MPI_Group_free(MPI_Group *group)
```

Input Parameter
group group (handle)

Notes

On output, group is set to `MPI_GROUP_NULL`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_ARG
: This error class is associated with an error code that indicates that an attempt was made to free one of the permanent groups.

Location

`group_free.c`

MPI_Group_incl **MPI_Group_incl**

MPI_Group_incl — Produces a group by reordering an existing group and taking only listed members

Synopsis

```
int MPI_Group_incl(MPI_Group group, int n, int *ranks, MPI_Group *newgroup)
```

Input Parameters

group group (handle)
n number of elements in array `ranks` (and size of newgroup) (integer)
ranks ranks of processes in `group` to appear in `newgroup` (array of integers)

Output Parameter

newgroup new group derived from above, in the order defined by `ranks` (handle)

Note

This implementation does not currently check to see that the list of ranks to ensure that there are no duplicates.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_GROUP
Null or invalid group passed to function.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

See Also

MPI_Group_free

Location

`group_incl.c`

MPI_Group_intersection **MPI_Group_intersection**

MPI_Group_intersection — Produces a group as the intersection of two existing groups

Synopsis

```
int MPI_Group_intersection(MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)
```

Input Parameters

group1 first group (handle)
group2 second group (handle)

Output Parameter

newgroup intersection group (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_GROUP
Null or invalid group passed to function.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

See Also

MPI_Group_free

Location

`group_intersection.c`

MPI_Group_range_excl **MPI_Group_range_excl**

MPI_Group_range_excl — Produces a group by excluding ranges of processes from an existing group

Synopsis

```
int MPI_Group_range_excl(MPI_Group group, int n, int ranges[][3], MPI_Group *newgroup)
```

Input Parameters

group group (handle)
n number of elements in array `ranks` (integer)
ranges a one-dimensional array of integer triplets of the form (first rank, last rank, stride), indicating the ranks in `group` of processes to be excluded from the output group `newgroup` .

Output Parameter

newgroup new group derived from above, preserving the order in `group` (handle)

Note

The MPI standard requires that each of the ranks to be excluded must be a valid rank in the group and all elements must be distinct or the function is erroneous.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_GROUP
: Null or invalid group passed to function.

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

See Also

MPI_Group_free

Location

`group_range_excl.c`

MPI_Group_range_incl **MPI_Group_range_incl**

MPI_Group_range_incl — Creates a new group from ranges of ranks in an existing group

Synopsis

```
int MPI_Group_range_incl(MPI_Group group, int n, int ranges[][3], MPI_Group *newgroup)
```

Input Parameters

group group (handle)
n number of triplets in array `ranges` (integer)
ranges a one-dimensional array of integer triplets, of the form (first rank, last rank, stride) indicating ranks in `group` or processes to be included in `newgroup`

Output Parameter

newgroup new group derived from above, in the order defined by `ranges` (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_GROUP
Null or invalid group passed to function.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

See Also

MPI_Group_free

Location

`group_range_incl.c`

MPI_Group_rank — Returns the rank of this process in the given group

Synopsis

```
int MPI_Group_rank(MPI_Group group, int *rank)
```

Input Parameters

group group (handle)

Output Parameter

rank rank of the calling process in group, or `MPI_UNDEFINED` if the process is not a member (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_GROUP
Null or invalid group passed to function.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`group_rank.c`

MPI_Group_size — Returns the size of a group

Synopsis

```
int MPI_Group_size(MPI_Group group, int *size)
```

Input Parameters

group group (handle) Output Parameter:
size number of processes in the group (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_GROUP
Null or invalid group passed to function.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`group_size.c`

MPI_Group_translate_ranks — group_translate_ranks

Synopsis

```
int MPI_Group_translate_ranks(MPI_Group group1, int n, int *ranks1, MPI_Group group2, int *ranks2)
```

Input Parameters

group1 group1 (handle)
n number of ranks in `ranks1` and `ranks2` arrays (integer)
ranks1 array of zero or more valid ranks in `group1`
group2 group2 (handle)

Output Parameter

ranks2 array of corresponding ranks in group2, `MPI_UNDEFINED` when no correspondence exists.

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`group_translate_ranks.c`

MPI_Group_union **MPI_Group_union**

MPI_Group_union — Produces a group by combining two groups

Synopsis

```
int MPI_Group_union(MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)
```

Input Parameters

group1 first group (handle)
group2 second group (handle)

Output Parameter

newgroup union group (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_GROUP
Null or invalid group passed to function.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

See Also

MPI_Group_free

Location

`group_union.c`

MPI_Ibsend — Starts a nonblocking buffered send

Synopsis

```
int MPI_Ibsend(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
               MPI_Comm comm, MPI_Request *request)
```

Input Parameters

buf initial address of send buffer (choice)
count number of elements in send buffer (integer)
datatype datatype of each send buffer element (handle)
dest rank of destination (integer)
tag message tag (integer)
comm communicator (handle)

Output Parameter

request communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_BUFFER
Invalid buffer pointer. Usually a null buffer where one is not valid.

Location

`ibsend.c`

MPI_Info_create **MPI_Info_create**

MPI_Info_create — Creates a new info object

Synopsis

```
int MPI_Info_create( MPI_Info *info )
```

Output Argument

info info object (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`info_create.c`

MPI_Info_delete **MPI_Info_delete**

MPI_Info_delete — Deletes a (key,value) pair from info

Synopsis

```
int MPI_Info_delete( MPI_Info info, char *key )
```

Input Parameters

info info object (handle)
key key (string)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`info_delete.c`

MPI_Info_dup **MPI_Info_dup**

MPI_Info_dup — Returns a duplicate of the info object

Synopsis

```
int MPI_Info_dup( MPI_Info info, MPI_Info *newinfo )
```

Input Arguments

info info object (handle)

Output Arguments

newinfo duplicate of info object (handle)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`info_dup.c`

MPI_Info_free — Frees an info object

Synopsis

```
int MPI_Info_free( MPI_Info *info )
```

Input Parameter

info info object (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_INFO
Invalid Info

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`info_free.c`

MPI_Info_get **MPI_Info_get**

MPI_Info_get — Retrieves the value associated with a key

Synopsis

```
int MPI_Info_get(MPI_Info info, char *key, int valuelen, char *value,
                 int *flag)
```

Input Parameters

info info object (handle)
key key (string)
valuelen length of value argument (integer)

Output Parameters

value value (string)
flag true if key defined, false if not (boolean)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_INFO_KEY
Invalid or null key string for info.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_INFO_VALUE
Invalid or null value string for info

Location

`info_get.c`

MPI_Info_get_nkeys **MPI_Info_get_nkeys**

MPI_Info_get_nkeys — Returns the number of currently defined keys in info

Synopsis

```
int MPI_Info_get_nkeys( MPI_Info info, int *nkeys )
```

Input Arguments

info info object (handle)

Output Arguments

nkeys number of defined keys (integer)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`info_getn.c`

MPI_Info_get_nthkey **MPI_Info_get_nthkey**

MPI_Info_get_nthkey — Returns the nth defined key in info

Synopsis

```
int MPI_Info_get_nthkey( MPI_Info info, int n, char *key )
```

Input Arguments

info info object (handle)
n key number (integer)

Output Argument

keys key (string). The maximum number of characters is `MPI_MAX_INFO_KEY`.

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`info_getnth.c`

MPI_Info_get_valuelen **MPI_Info_get_valuelen**

MPI_Info_get_valuelen — Retrieves the length of the value associated with a key

Synopsis

```
int MPI_Info_get_valuelen( MPI_Info info, char *key, int *valuelen, int *flag )
```

Input Arguments

info info object (handle)
key key (string)

Output Arguments

valuelen length of value argument (integer)
flag true if key defined, false if not (boolean)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_INFO_KEY
Invalid or null key string for info.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`info_getvallen.c`

MPI_Info_set **MPI_Info_set**

MPI_Info_set — Adds a (key,value) pair to info

Synopsis

```
int MPI_Info_set( MPI_Info info, char *key, char *value )
```

Input Parameters

info	info object (handle)
key	key (string)
value	value (string)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_INFO_KEY
Invalid or null key string for info.

MPI_ERR_INFO_VALUE
Invalid or null value string for info

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

`info_set.c`

MPI_Init **MPI_Init**

MPI_Init — Initialize the MPI execution environment

Synopsis

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

Input Parameters

argc Pointer to the number of arguments
argv Pointer to the argument vector

Notes

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`init.c`

MPI_Init_thread **MPI_Init_thread**

MPI_Init_thread — Initialize the MPI execution environment

Synopsis

```
int MPI_Init_thread( int *argc, char ***argv, int required, int *provided )
```

Input Parameters

argc Pointer to the number of arguments
argv Pointer to the argument vector
required Level of desired thread support

Output Parameter

provided Level of provided thread support

Command line arguments

MPI specifies no command-line arguments but does allow an MPI implementation to make use of them. See `MPI_INIT` for a description of the command line arguments supported by `MPI_INIT` and `MPI_INIT_THREAD`.

Notes

Note that the Fortran binding for this routine does not have the `argc` and `argv` arguments. (`MPI_INIT_THREAD(required, provided, ierror)`)
The valid values for the level of thread support are:

MPI_THREAD_SINGLE
Only one thread will execute.

MPI_THREAD_FUNNELED
The process may be multi-threaded, but only the main thread will make MPI calls (all MPI calls are funneled to the main thread).

MPI_THREAD_SERIALIZED
The process may be multi-threaded, and multiple threads may make MPI calls, but only one at a time: MPI calls are not made concurrently from two distinct threads (all MPI calls are serialized).

MPI_THREAD_MULTIPLE
Multiple threads may call MPI, with no restrictions.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

initthread.c

MPI_Initialized **MPI_Initialized**

MPI_Initialized — Indicates whether MPI_Init has been called.

Synopsis

```
int MPI_Initialized( int *flag )
```

Output Argument

flag Flag is true if MPI_Init or MPI_Init_thread has been called and false otherwise.

Notes

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

initialized.c

MPI_Intercomm_create **MPI_Intercomm_create**

MPI_Intercomm_create — Creates an intercommuncator from two intracommunicators

Synopsis

```
int MPI_Intercomm_create(MPI_Comm local_comm, int local_leader,
                         MPI_Comm peer_comm, int remote_leader, int tag,
                         MPI_Comm *newintercomm)
```

Input Paramters

local_comm Local (intra)communicator
local_leader Rank in local_comm of leader (often 0)
peer_comm Remote communicator
remote_leader
Rank in peer_comm of remote leader (often 0)

tag Message tag to use in constructing intercommunicator; if multiple `MPI_Intercomm_creates` are being made, they should use different tags (more precisely, ensure that the local and remote leaders are using different tags for each `MPI_intercomm_create`).

Output Parameter

comm_out Created intercommunicator

Notes

The MPI 1.1 Standard contains two mutually exclusive comments on the input intracommunicators. One says that their repective groups must be disjoint; the other that the leaders can be the same process. After some discussion by the MPI Forum, it has been decided that the groups must be disjoint. Note that the *reason* given for this in the standard is *not* the reason for this choice; rather, the *other* operations on intercommunicators (like `MPI_Intercomm_merge`) do not make sense if the groups are not disjoint.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS

No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

See Also

MPI_Intercomm_merge, MPI_Comm_free, MPI_Comm_remote_group, MPI_Comm_remote_size

Notes

`peer_comm` is significant only for the process designated the `local_leader` in the `local_comm`.

Location

`intercomm_create.c`

MPI_Intercomm_merge **MPI_Intercomm_merge**

MPI_Intercomm_merge — Creates an intracommuncator from an intercommunicator

Synopsis

```
int MPI_Intercomm_merge(MPI_Comm intercomm, int high, MPI_Comm *newintracomm)
```

Input Parameters

comm Intercommunicator

high Used to order the groups of the two intracommunicators within comm when creating the new communicator. This is a boolean value; the group that sets high true has its processes ordered *after* the group that sets this value to false.

Output Parameter

comm_out Created intracommunicator

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Algorithm

```
1) Allocate contexts
2) Local and remote group leaders swap high values
3) Determine the high value.
4) Merge the two groups and make the intra-communicator
```

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

See Also

MPI_Intercomm_create, MPI_Comm_free
MPI_Intercomm_merge - merge communicators

Location

`intercomm_merge.c`

MPI_Iprobe — Nonblocking test for a message

Synopsis

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

Input Parameters

source source rank, or `MPI_ANY_SOURCE` (integer)
tag tag value or `MPI_ANY_TAG` (integer)
comm communicator (handle)

Output Parameter

flag (logical)
status status object (Status)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

Location

iprobe.c

MPI_Irecv **MPI_Irecv**

MPI_Irecv — Begins a nonblocking receive

Synopsis

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

Input Parameters

buf	initial address of receive buffer (choice)
count	number of elements in receive buffer (integer)
datatype	datatype of each receive buffer element (handle)
source	rank of source (integer)
tag	message tag (integer)
comm	communicator (handle)

Output Parameter

request	communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Location

irecv.c

MPI_Irsend **MPI_Irsend**

MPI_Irsend — Starts a nonblocking ready send

Synopsis

```
int MPI_Irsend(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
               MPI_Comm comm, MPI_Request *request)
```

Input Parameters

buf initial address of send buffer (choice)
count number of elements in send buffer (integer)
datatype datatype of each send buffer element (handle)
dest rank of destination (integer)
tag message tag (integer)
comm communicator (handle) Output Parameter:
request communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

irsend.c

MPI_Is_thread_main **MPI_Is_thread_main**

MPI_Is_thread_main — Returns a flag indicating whether this thread called MPI_Init or MPI_Init_thread

Synopsis

```
int MPI_Is_thread_main( int *flag )
```

Output Arguments

flag Flag is true if MPI_Init or MPI_Init_thread has been called by this thread and false otherwise.

Notes

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

ismain.c

MPI_Isend **MPI_Isend**

MPI_Isend — Begins a nonblocking send

Synopsis

```
int MPI_Isend(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
              MPI_Comm comm, MPI_Request *request)
```

Input Parameters

buf initial address of send buffer (choice)
count number of elements in send buffer (integer)
datatype datatype of each send buffer element (handle)
dest rank of destination (integer)
tag message tag (integer)
comm communicator (handle)

Output Parameter

request communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

`isend.c`

MPI_Issend **MPI_Issend**

MPI_Issend — Starts a nonblocking synchronous send

Synopsis

```
int MPI_Issend(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
               MPI_Comm comm, MPI_Request *request)
```

Input Parameters

buf	initial address of send buffer (choice)
count	number of elements in send buffer (integer)
datatype	datatype of each send buffer element (handle)
dest	rank of destination (integer)
tag	message tag (integer)
comm	communicator (handle)

Output Parameter

request	communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler

MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in MPI_Comm_rank).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see MPI_Type_commit).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (MPI_Recv, MPI_Irecv, MPI_Sendrecv, etc.) may also be MPI_ANY_TAG. The largest tag value is available through the the attribute MPI_TAG_UB.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (MPI_Recv, MPI_Irecv, MPI_Sendrecv, etc.) may also be MPI_ANY_SOURCE.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

issend.c

MPI_Keyval_create **MPI_Keyval_create**

MPI_Keyval_create — Generates a new attribute key

Synopsis

```
int MPI_Keyval_create(MPI_Copy_function *copy_fn,
                      MPI_Delete_function *delete_fn,
                      int *keyval, void *extra_state)
```

Input Parameters

copy_fn Copy callback function for keyval
delete_fn Delete callback function for keyval
extra_state Extra state for callback functions

Output Parameter

keyval key value for future access (integer)

Notes

Key values are global (available for any and all communicators).
There are subtle differences between C and Fortran that require that the copy_fn be written in the same language that `MPI_Keyval_create` is called from. This should not be a problem for most users; only programers using both Fortran and C in the same program need to be sure that they follow this rule.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`keyval_create.c`

MPI_Keyval_free **MPI_Keyval_free**

MPI_Keyval_free — Frees attribute key for communicator cache attribute

Synopsis

```
int MPI_Keyval_free(int *keyval)
```

Input Parameter

keyval Frees the integer key value (integer)

Note

Key values are global (they can be used with any and all communicators)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_ARG
: This error class is associated with an error code that indicates that an attempt was made to free one of the permanent keys.

See Also

MPI_Keyval_create

Location

`keyval_free.c`

MPI_Lookup_name **MPI_Lookup_name**

MPI_Lookup_name — Lookup a port given a service name

Synopsis

```
int MPI_Lookup_name(char *service_name, MPI_Info info, char *port_name)
```

Input Parameters

service_name a service name (string)
info implementation-specific information (handle)

Output Parameter

port_name a port name (string)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_INFO
Invalid Info

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`lookup_name.c`

MPI_Op_create — Creates a user-defined combination function handle

Synopsis

```
int MPI_Op_create(MPI_User_function *function, int commute, MPI_Op *op)
```

Input Parameters

function user defined function (function)
commute true if commutative; false otherwise.

Output Parameter

op operation (handle)

Notes on the user function

The calling list for the user function type is

```
typedef void (MPI_User_function) ( void * a,
              void * b, int * len, MPI_Datatype * );
```

where the operation is `b[i] = a[i] op b[i]`, for `i=0,...,len-1`. A pointer to the datatype given to the MPI collective computation routine (i.e., `MPI_Reduce`, `MPI_Allreduce`, `MPI_Scan`, or `MPI_Reduce_scatter`) is also passed to the user-specified routine.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Notes on collective operations

The reduction functions (`MPI_Op`) do not return an error value. As a result, if the functions detect an error, all they can do is either call `MPI_Abort` or silently skip the problem. Thus, if you change the error handler from `MPI_ERRORS_ARE_FATAL` to something else, for example, `MPI_ERRORS_RETURN`, then no error may be indicated.
The reason for this is the performance problems in ensuring that all collective routines return the same error value.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

See Also

MPI_Op_free

Location

`op_create.c`

MPI_Op_free **MPI_Op_free**

MPI_Op_free — Frees a user-defined combination function handle

Synopsis

```
int MPI_Op_free(MPI_Op *op)
```

Input Parameter

op operation (handle)

Notes

`op` is set to `MPI_OP_NULL` on exit.

Null Handles

The MPI 1.1 specification, in the section on opaque objects, explicitly

disallows freeing a null communicator. The text from the standard is

```
A null handle argument is an erroneous IN argument in MPI calls, unless an
exception is explicitly stated in the text that defines the function. Such
exception is allowed for handles to request objects in Wait and Test calls
(sections Communication Completion and Multiple Completions ). Otherwise, a
null handle can only be passed to a function that allocates a new object and
returns a reference to it in the handle.
```

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_ARG
: Invalid argument; the error code associated with this error indicates an attempt to free an MPI permanent operation (e.g., `MPI_SUM`). *N/ /*N MPI_ERR_PERM_KEY

MPI_ERR_ARG
: Invalid argument; the error code associated with this error indicates an attempt to free or chnage an MPI permanent keyval (e.g., `MPI_TAG_UB`). *N/ /*N MPI_ERR_UNKNOWN

MPI_ERR_UNKNOWN
: Unknown error. You should never see this. If you do, report it to `mpi-bugs@mcs.anl.gov`.

See Also

MPI_Op_create

Location

`op_free.c`

MPI_Open_port **MPI_Open_port**

MPI_Open_port — short description

Synopsis

```
int MPI_Open_port(MPI_Info info, char *port_name)
```

Input Parameter

info implementation-specific information on how to establish an address (handle)

Output Parameter

port_name newly established port (string)

Notes

MPI copies a system-supplied port name into `port_name`. `port_name` identifies the newly opened port and can be used by a client to contact the server. The maximum size string that may be supplied by the system is `MPI_MAX_PORT_NAME`.

Reserved Info Key Values

ip_port Value contains IP port number at which to establish a port.
ip_address Value contains IP address at which to establish a port. If the address is not a valid IP address of the host on which the `MPI_OPEN_PORT` call is made, the results are undefined.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`open_port.c`

MPI_Pack_external **MPI_Pack_external**

MPI_Pack_external — pack external

Synopsis

```
int MPI_Pack_external(char *datarep,
                      void *inbuf,
                      int incount,
```

```
MPI_Datatype datatype,
void *outbuf,
MPI_Aint outcount,
MPI_Aint *position)
```

Input Parameters

datarep data representation (string)
inbuf input buffer start (choice)
incount number of input data items (integer)
datatype datatype of each input data item (handle)
outsize output buffer size, in bytes (integer)

Output Parameter

outbuf output buffer start (choice)

Input/Output Parameter

position current position in buffer, in bytes (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`pack_external.c`

MPI_Pack_external_size **MPI_Pack_external_size**

MPI_Pack_external_size — pack external size

Synopsis

```
int MPI_Pack_external_size(char *datarep,
                           int incount,
                           MPI_Datatype datatype,
                           MPI_Aint *size)
```

Input Parameters

datarep data representation (string)
incount number of input data items (integer)
datatype datatype of each input data item (handle)

Output Parameters

size output buffer size, in bytes (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

pack_external_size.c

MPI_Pack_size **MPI_Pack_size**

MPI_Pack_size — Returns the upper bound on the amount of space needed to pack a message

Synopsis

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

Input Parameters

incount	count argument to packing call (integer)
datatype	datatype argument to packing call (handle)
comm	communicator argument to packing call (handle)

Output Parameter

size	upper bound on size of packed message, in bytes (integer)

Notes

The MPI standard document describes this in terms of MPI_Pack, but it applies to both MPI_Pack and MPI_Unpack. That is, the value size is the maximum that is needed by either MPI_Pack or MPI_Unpack.

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`pack_size.c`

MPI_Pcontrol **MPI_Pcontrol**

MPI_Pcontrol — Controls profiling

Synopsis

```
int MPI_Pcontrol(const int level, ...)
```

Input Parameters

level Profiling level
... other arguments

Notes

This routine provides a common interface for profiling control. The interpretation of `level` and any other arguments is left to the profiling library.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine

MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

pcontrol.c

MPI_Probe **MPI_Probe**

MPI_Probe — Blocking test for a message

Synopsis

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

Input Parameters

source source rank, or MPI_ANY_SOURCE (integer)
tag tag value or MPI_ANY_TAG (integer)
comm communicator (handle)

Output Parameter

status status object (Status)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.

All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS

No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

Location

`probe.c`

MPI_Publish_name **MPI_Publish_name**

MPI_Publish_name — Publish a service name for use with MPI_Comm_connect

Synopsis

```
int MPI_Publish_name(char *service_name, MPI_Info info, char *port_name)
```

Input Parameters

service_name a service name to associate with the port (string)
info implementation-specific information (handle)
port_name a port name (string)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_INFO
Invalid Info

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`publish_name.c`

MPI_Put **MPI_Put**

MPI_Put — put

Synopsis

```
int MPI_Put(void *origin_addr, int origin_count, MPI_Datatype
            origin_datatype, int target_rank, MPI_Aint target_disp,
            int target_count, MPI_Datatype target_datatype, MPI_Win
            win)
```

Input Parameters

origin_addr initial address of origin buffer (choice)
origin_count number of entries in origin buffer (nonnegative integer)
origin_datatype
datatype of each entry in origin buffer (handle)

target_rank rank of target (nonnegative integer)
target_disp displacement from start of window to target buffer (nonnegative integer)
target_count number of entries in target buffer (nonnegative integer)
target_datatype
datatype of each entry in target buffer (handle)

win window object used for communication (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_WIN
: Invalid MPI window object

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`put.c`

MPI_Query_thread — Return the level of thread support provided

Synopsis

```
int MPI_Query_thread( int *provided )
```

Output Parameter

provided Level of thread support provided. This is the same value that was returned in the `provided` argument in `MPI_Init_thread`.

Notes

If `MPI_Init` was called instead of `MPI_Init_thread`, the level of thread support is defined by the implementation. This routine allows you to find out the provided level. It is also useful for library routines that discover that MPI has already been initialized and wish to determine what level of thread support is available.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`querythread.c`

MPI_Recv — Basic receive

Synopsis

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

Output Parameters

buf	initial address of receive buffer (choice)
status	status object (Status)

Input Parameters

count	maximum number of elements in receive buffer (integer)
datatype	datatype of each receive buffer element (handle)
source	rank of source (integer)
tag	message tag (integer)
comm	communicator (handle)

Notes

The `count` argument indicates the maximum length of a message; the actual number can be determined with `MPI_Get_count`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TAG
: Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

Location

`recv.c`

MPI_Recv_init **MPI_Recv_init**

MPI_Recv_init — Builds a handle for a receive

Synopsis

```
int MPI_Recv_init(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MP
```

Input Parameters

buf initial address of receive buffer (choice)
count number of elements received (integer)
datatype type of each element (handle)
source rank of source or `MPI_ANY_SOURCE` (integer)
tag message tag or `MPI_ANY_TAG` (integer)
comm communicator (handle)

Output Parameter

request communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

See Also

MPI_Start, MPI_Request_free

Location

`recv_init.c`

MPI_Reduce **MPI_Reduce**

MPI_Reduce — Reduces values on all processes to a single value

Synopsis

```
int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, M
```

Input Parameters

sendbuf	address of send buffer (choice)
count	number of elements in send buffer (integer)
datatype	data type of elements of send buffer (handle)
op	reduce operation (handle)
root	rank of root process (integer)
comm	communicator (handle)

Output Parameter

recvbuf	address of receive buffer (choice, significant only at `root`)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Notes on collective operations

The reduction functions (`MPI_Op`) do not return an error value. As a result, if the functions detect an error, all they can do is either call `MPI_Abort` or silently skip the problem. Thus, if you change the error handler from `MPI_ERRORS_ARE_FATAL` to something else, for example, `MPI_ERRORS_RETURN`, then no error may be indicated.
The reason for this is the performance problems in ensuring that all collective routines return the same error value.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
: Invalid buffer pointer. Usually a null buffer where one is not valid.

MPI_ERR_BUFFER
: This error class is associcated with an error code that indicates that two buffer arguments are *aliased*; that is, the describe overlapping storage (often the exact same storage). This is prohibited in MPI (because it is prohibited by the Fortran standard, and rather than have a separate case for C and Fortran, the MPI Forum adopted the more restrictive requirements of Fortran).

Location

`reduce.c`

MPI_Reduce_scatter **MPI_Reduce_scatter**

MPI_Reduce_scatter — Combines values and scatters the results

Synopsis

```
int MPI_Reduce_scatter(void *sendbuf, void *recvbuf, int *recvcnts, MPI_Datatype datatype, MPI_Op op,
```

Input Parameters

sendbuf starting address of send buffer (choice)

recvcounts integer array specifying the number of elements in result distributed to each process. Array must be identical on all calling processes.

datatype data type of elements of input buffer (handle)

op operation (handle)

comm communicator (handle)

Output Parameter

recvbuf starting address of receive buffer (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Notes on collective operations

The reduction functions (`MPI_Op`) do not return an error value. As a result, if the functions detect an error, all they can do is either call `MPI_Abort` or silently skip the problem. Thus, if you change the error handler from `MPI_ERRORS_ARE_FATAL` to something else, for example, `MPI_ERRORS_RETURN`, then no error may be indicated.
The reason for this is the performance problems in ensuring that all collective routines return the same error value.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
Invalid buffer pointer. Usually a null buffer where one is not valid.

MPI_ERR_OP
Invalid operation. MPI operations (objects of type `MPI_Op`) must either be one of the predefined operations (e.g., `MPI_SUM`) or created with `MPI_Op_create`.

MPI_ERR_BUFFER

This error class is associcated with an error code that indicates that two buffer arguments are *aliased*; that is, the describe overlapping storage (often the exact same storage). This is prohibited in MPI (because it is prohibited by the Fortran standard, and rather than have a separate case for C and Fortran, the MPI Forum adopted the more restrictive requirements of Fortran).

Location

`red_scat.c`

MPI_Register_datarep **MPI_Register_datarep**

MPI_Register_datarep — register datarep

Synopsis

```
int MPI_Register_datarep(char *datarep, MPI_Datarep_conversion_function *read_conversion_fn, MPI_Data
```

Input Parameters

datarep data representation identifier (string)

read_conversion_fn
function invoked to convert from file representation to native representation (function)

write_conversion_fn
function invoked to convert from native representation to file representation (function)

dtype_file_extent_fn
function invoked to get the extent of a datatype as represented in the file (function)

extra_state extra state

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not*

guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`register_datarep.c`

MPI_Request_free **MPI_Request_free**

MPI_Request_free — Frees a communication request object

Synopsis

```
int MPI_Request_free(MPI_Request *request)
```

Input Parameter

request communication request (handle)

Notes

This routine is normally used to free inactive persistent requests created with either `MPI_Recv_init` or `MPI_Send_init` and friends. It *is* also permissible to free an active request. However, once freed, the request can no longer be used in a wait or test routine (e.g., `MPI_Wait`) to determine completion. This routine may also be used to free a non-persistent requests such as those created with `MPI_Irecv` or `MPI_Isend` and friends. Like active persistent requests, once freed, the request can no longer be used with test/wait routines to determine completion.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler

`MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_REQUEST
Invalid `MPI_Request`. Either null or, in the case of a `MPI_Start` or `MPI_Startall`, not a persistent request.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

See Also

also: MPI_Isend, MPI_Irecv, MPI_Issend, MPI_Ibsend, MPI_Irsend, MPI_Recv_init, MPI_Send_init, MPI_Ssend_init, MPI_Rsend_init, MPI_Wait, MPI_Test, MPI_Waitall, MPI_Waitany, MPI_Waitsome, MPI_Testall, MPI_Testany, MPI_Testsome

Location

`request_free.c`

MPI_Request_get_status **MPI_Request_get_status**

MPI_Request_get_status — Nondestructive test for the completion of a Request

Synopsis

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

Input Parameter

MPI_Request request
request handle

Output Parameters

int *flag true if operation has completed (logical)

MPI_Status *status
status object (Status). May be `MPI_STATUS_IGNORE`.

Notes

Unlike MPI_Test, MPI_Request_get_status does not deallocate or deactivate the request. A call to one of the test/wait routines or MPI_Request_free should be made to release the request object.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`request_get_status.c`

MPI_Rsend — Basic ready send

Synopsis

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

Input Parameters

buf initial address of send buffer (choice)
count number of elements in send buffer (nonnegative integer)
datatype datatype of each send buffer element (handle)
dest rank of destination (integer)
tag message tag (integer)
comm communicator (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in MPI_Comm_rank).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see MPI_Type_commit).

MPI_ERR_TAG
: Invalid tag argument. Tags must be non-negative; tags in a receive (MPI_Recv, MPI_Irecv, MPI_Sendrecv, etc.) may also be MPI_ANY_TAG. The largest tag value is available through the the attribute MPI_TAG_UB.

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (MPI_Recv, MPI_Irecv, MPI_Sendrecv, etc.) may also be MPI_ANY_SOURCE.

Location

rsend.c

MPI_Rsend_init **MPI_Rsend_init**

MPI_Rsend_init — Builds a handle for a ready send

Synopsis

```
int MPI_Rsend_init(void *buf, int count, MPI_Datatype datatype, int dest,
                   int tag, MPI_Comm comm, MPI_Request *request)
```

Input Parameters

buf initial address of send buffer (choice)
count number of elements sent (integer)

datatype type of each element (handle)
dest rank of destination (integer)
tag message tag (integer)
comm communicator (handle)

Output Parameter

request communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

See Also

MPI_Start, MPI_Request_free, MPI_Send_init

Location

`rsend_init.c`

MPI_Scan **MPI_Scan**

MPI_Scan — Computes the scan (partial reductions) of data on a collection of processes

Synopsis

```
int MPI_Scan(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm
```

Input Parameters

sendbuf	starting address of send buffer (choice)
count	number of elements in input buffer (integer)
datatype	data type of elements of input buffer (handle)
op	operation (handle)
comm	communicator (handle)

Output Parameter

recvbuf	starting address of receive buffer (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Notes on collective operations

The reduction functions (`MPI_Op`) do not return an error value. As a result, if the functions detect an error, all they can do is either call `MPI_Abort` or silently skip the problem. Thus, if you change the error handler from `MPI_ERRORS_ARE_FATAL` to something else, for example, `MPI_ERRORS_RETURN`, then no error may be indicated.
The reason for this is the performance problems in ensuring that all collective routines return the same error value.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
: Invalid buffer pointer. Usually a null buffer where one is not valid.

MPI_ERR_BUFFER
: This error class is associcated with an error code that indicates that two buffer arguments are *aliased*; that is, the describe overlapping storage (often the exact same storage). This is prohibited in MPI (because it is prohibited by the Fortran standard, and rather than have a separate case for C and Fortran, the MPI Forum adopted the more restrictive requirements of Fortran).

Location

`scan.c`

MPI_Scatter — Sends data from one task to all other tasks in a group

Synopsis

```
int MPI_Scatter(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf, int recvcnt, MPI_Da
```

Input Parameters

sendbuf address of send buffer (choice, significant only at `root`)
sendcount number of elements sent to each process (integer, significant only at `root`)
sendtype data type of send buffer elements (significant only at `root`) (handle)
recvcount number of elements in receive buffer (integer)
recvtype data type of receive buffer elements (handle)

root rank of sending process (integer)
comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
Invalid buffer pointer. Usually a null buffer where one is not valid.

Location

`scatter.c`

MPI_Scatterv **MPI_Scatterv**

MPI_Scatterv — Scatters a buffer in parts to all tasks in a group

Synopsis

```
int MPI_Scatterv( void *sendbuf, int *sendcnts, int *displs, MPI_Datatype sendtype, void *recvbuf, in
```

```
              int root, MPI_Comm comm)
```

Input Parameters

sendbuf address of send buffer (choice, significant only at `root`)
sendcounts integer array (of length group size) specifying the number of elements to send to each processor
displs integer array (of length group size). Entry `i` specifies the displacement (relative to sendbuf from which to take the outgoing data to process `i`
sendtype data type of send buffer elements (handle)
recvcount number of elements in receive buffer (integer)
recvtype data type of receive buffer elements (handle)
root rank of sending process (integer)
comm communicator (handle)

Output Parameter

recvbuf address of receive buffer (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_BUFFER
Invalid buffer pointer. Usually a null buffer where one is not valid.

Location

scatterv.c

MPI_Send **MPI_Send**

MPI_Send — Performs a basic send

Synopsis

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

Input Parameters

buf	initial address of send buffer (choice)
count	number of elements in send buffer (nonnegative integer)
datatype	datatype of each send buffer element (handle)
dest	rank of destination (integer)
tag	message tag (integer)
comm	communicator (handle)

Notes

This routine may block until the message is received.

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in MPI_Comm_rank).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

See Also

MPI_Isend, MPI_Bsend

Location

`send.c`

MPI_Send_init **MPI_Send_init**

MPI_Send_init — Builds a handle for a standard send

Synopsis

```
int MPI_Send_init(void *buf, int count, MPI_Datatype datatype, int dest,
                  int tag, MPI_Comm comm, MPI_Request *request)
```

Input Parameters

buf initial address of send buffer (choice)
count number of elements sent (integer)
datatype type of each element (handle)
dest rank of destination (integer)
tag message tag (integer)
comm communicator (handle) Output Parameter:
request communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_TAG
: Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

See Also

MPI_Start, MPI_Startall, MPI_Request_free

Location

`send_init.c`

MPI_Sendrecv **MPI_Sendrecv**

MPI_Sendrecv — Sends and receives a message

Synopsis

```
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)
```

Input Parameters

sendbuf initial address of send buffer (choice)
sendcount number of elements in send buffer (integer)
sendtype type of elements in send buffer (handle)
dest rank of destination (integer)
sendtag send tag (integer)
recvcount number of elements in receive buffer (integer)
recvtype type of elements in receive buffer (handle)
source rank of source (integer)
recvtag receive tag (integer)
comm communicator (handle)

Output Parameters

recvbuf initial address of receive buffer (choice)
status status object (Status). This refers to the receive operation.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK

Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

Location

`sendrecv.c`

MPI_Sendrecv_replace **MPI_Sendrecv_replace**

MPI_Sendrecv_replace — Sends and receives using a single buffer

Synopsis

```
int MPI_Sendrecv_replace(void *buf, int count, MPI_Datatype datatype, int dest, int sendtag, int sou
                         MPI_Comm comm, MPI_Status *status)
```

Input Parameters

count	number of elements in send and receive buffer (integer)
datatype	type of elements in send and receive buffer (handle)
dest	rank of destination (integer)
sendtag	send message tag (integer)
source	rank of source (integer)
recvtag	receive message tag (integer)
comm	communicator (handle)

Output Parameters

buf	initial address of send and receive buffer (choice)
status	status object (Status)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not*

guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_TAG
: Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

MPI_ERR_TRUNCATE
: Message truncated on receive. The buffer size specified was too small for the received message. This is a recoverable error in the MPICH implementation.

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

`sendrecv_rep.c`

MPI_Ssend **MPI_Ssend**

MPI_Ssend — Basic synchronous send

Synopsis

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

Input Parameters

buf	initial address of send buffer (choice)
count	number of elements in send buffer (nonnegative integer)
datatype	datatype of each send buffer element (handle)
dest	rank of destination (integer)
tag	message tag (integer)
comm	communicator (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_COMM
: Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_TAG
: Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
: Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

Location

`ssend.c`

MPI_Ssend_init **MPI_Ssend_init**

MPI_Ssend_init — Builds a handle for a synchronous send

Synopsis

```
int MPI_Ssend_init(void *buf, int count, MPI_Datatype datatype, int dest,
                   int tag, MPI_Comm comm, MPI_Request *request)
```

Input Parameters

buf initial address of send buffer (choice)
count number of elements sent (integer)
datatype type of each element (handle)
dest rank of destination (integer)
tag message tag (integer)
comm communicator (handle)

Output Parameter

request communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_TAG
Invalid tag argument. Tags must be non-negative; tags in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_TAG`. The largest tag value is available through the the attribute `MPI_TAG_UB`.

MPI_ERR_RANK
Invalid source or destination rank. Ranks must be between zero and the size of the communicator minus one; ranks in a receive (`MPI_Recv`, `MPI_Irecv`, `MPI_Sendrecv`, etc.) may also be `MPI_ANY_SOURCE`.

Location

ssend_init.c

MPI_Start **MPI_Start**

MPI_Start — Initiates a communication with a persistent request handle

Synopsis

```
int MPI_Start(MPI_Request *request)
```

Input Parameter

request communication request (handle)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_REQUEST
Invalid MPI_Request. Either null or, in the case of a MPI_Start or MPI_Startall, not a persistent request.

Location

start.c

MPI_Startall **MPI_Startall**

MPI_Startall — Starts a collection of requests

Synopsis

```
int MPI_Startall(int count, MPI_Request array_of_requests[])
```

Input Parameters

count list length (integer)
array_of_requests
array of requests (array of handle)

Notes

Unlike MPI_Waitall(), MPI_Startall() does not provide a mechanism for returning multiple errors nor pinpointing the request(s) involved. Futhermore, the behavior of MPI_Startall() after an error occurs is not defined by the MPI standard. If well defined error reporting and behavior are required, multiple calls to MPI_Start() should be used instead.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`startall.c`

MPI_Status_set_cancelled — Sets the cancelled state associated with a Status object

Synopsis

```
int MPI_Status_set_cancelled(MPI_Status *status, int flag)
```

Input Parameters

MPI_Status *status
: status to associate cancel flag with (Status)

int flag
: if true indicates request was cancelled (logical)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`status_set_cancelled.c`

MPI_Status_set_elements — status set elements

Synopsis

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

Input Parameters

status status to associate count with (Status)
datatype datatype associated with count (handle)
count number of elements to associate with status (integer)

Arguments

MPI_Status *status
status

MPI_Datatype datatype
datatype

int count count

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`status_set_elements.c`

MPI_Test **MPI_Test**

MPI_Test — Tests for the completion of a send or receive

Synopsis

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

Input Parameter

request communication request (handle)

Output Parameter

flag true if operation completed (logical)
status status object (Status). May be `MPI_STATUS_IGNORE`.

Note on status for send operations

For send operations, the only use of status is for `MPI_Test_cancelled` or in the case that there is an error, in which case the `MPI_ERROR` field of status will be set.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_REQUEST
Invalid `MPI_Request`. Either null or, in the case of a `MPI_Start` or `MPI_Startall`, not a persistent request.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`test.c`

MPI_Test_cancelled **MPI_Test_cancelled**

MPI_Test_cancelled — Tests to see if a request was cancelled

Synopsis

```
int MPI_Test_cancelled(MPI_Status *status, int *flag)
```

Input Parameter

status status object (Status)

Output Parameter

flag (logical)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Location

`test_cancelled.c`

MPI_Testall **MPI_Testall**

MPI_Testall — Tests for the completion of all previously initiated communications

Synopsis

```
int MPI_Testall(int count, MPI_Request array_of_requests[], int *flag, MPI_Status array_of_statuses[]
```

Input Parameters

count lists length (integer)
array_of_requests
array of requests (array of handles)

Output Parameters

flag (logical)

array_of_statuses
array of status objects (array of Status). May be `MPI_STATUSES_IGNORE`.

Notes

`flag` is true only if all requests have completed. Otherwise, flag is false and neither the `array_of_requests` nor the `array_of_statuses` is modified.
If one or more of the requests completes with an error, MPI_ERR_IN_STATUS is returned. An error value will be present is elements of array_of_status associated with the requests. Likewise, the MPI_ERROR field in the status elements associated with requests that have successfully completed will be MPI_SUCCESS. Finally, those requests that have not completed will have a value of MPI_ERR_PENDING.
While it is possible to list a request handle more than once in the array_of_requests, such an action is considered erroneous and may cause the program to unexecpectedly terminate or produce incorrect results.

Note on status for send operations

For send operations, the only use of status is for `MPI_Test_cancelled` or in the case that there is an error, in which case the `MPI_ERROR` field of status will be set.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_IN_STATUS
The actual error value is in the `MPI_Status` argument. This error class is returned only from the multiple-completion routines (`MPI_Testall`, `MPI_Testany`, `MPI_Testsome`, `MPI_Waitall`, `MPI_Waitany`, and `MPI_Waitsome`). The field `MPI_ERROR` in the status argument contains the error value or `MPI_SUCCESS` (no error and complete) or `MPI_ERR_PENDING` to indicate that the request has not completed.

The MPI Standard does not specify what the result of the multiple completion routines is when an error occurs. For example, in an `MPI_WAITALL`, does the routine wait for all requests to either fail or complete, or does it return immediately (with the MPI definition of immediately, which means independent of actions of other MPI processes)? MPICH has chosen to make the return immediate (alternately, local in MPI terms), and to use the error class `MPI_ERR_PENDING` (introduced in MPI 1.1) to indicate which requests have not completed. In most cases, only one request with an error will be detected in each call to an MPI routine that tests multiple requests. The requests that have not been processed (because an error occured in one of the requests) will have their `MPI_ERROR` field marked with `MPI_ERR_PENDING`.

MPI_ERR_REQUEST
: Invalid `MPI_Request`. Either null or, in the case of a `MPI_Start` or `MPI_Startall`, not a persistent request.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`testall.c`

MPI_Testany **MPI_Testany**

MPI_Testany — Tests for completion of any previdously initiated communication

Synopsis

```
int MPI_Testany(int count, MPI_Request array_of_requests[], int *index, int *flag, MPI_Status *status
```

Input Parameters

count list length (integer)

array_of_requests
: array of requests (array of handles)

Output Parameters

index index of operation that completed, or `MPI_UNDEFINED` if none completed (integer)

flag true if one of the operations is complete (logical)

status status object (Status). May be `MPI_STATUS_IGNORE`.

Notes

While it is possible to list a request handle more than once in the array_of_requests, such an action is considered erroneous and may cause the program to unexecpectedly terminate or produce incorrect results.

Note on status for send operations

For send operations, the only use of status is for `MPI_Test_cancelled` or in the case that there is an error, in which case the `MPI_ERROR` field of status will be set.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`testany.c`

MPI_Testsome **MPI_Testsome**

MPI_Testsome — Tests for some given communications to complete

Synopsis

```
int MPI_Testsome(int incount, MPI_Request array_of_requests[], int *outcount, int array_of_indices[],
```

Input Parameters

incount length of array_of_requests (integer)

array_of_requests
: array of requests (array of handles)

Output Parameters

outcount number of completed requests (integer)

array_of_indices
array of indices of operations that completed (array of integers)

array_of_statuses
array of status objects for operations that completed (array of Status). May be `MPI_STATUSES_IGNORE`.

Notes

While it is possible to list a request handle more than once in the array_of_requests, such an action is considered erroneous and may cause the program to unexecpectedly terminate or produce incorrect results.

Note on status for send operations

For send operations, the only use of status is for `MPI_Test_cancelled` or in the case that there is an error, in which case the `MPI_ERROR` field of status will be set.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_IN_STATUS
The actual error value is in the `MPI_Status` argument. This error class is returned only from the multiple-completion routines (`MPI_Testall`, `MPI_Testany`, `MPI_Testsome`, `MPI_Waitall`, `MPI_Waitany`, and `MPI_Waitsome`). The field `MPI_ERROR` in the status argument contains the error value or `MPI_SUCCESS` (no error and complete) or `MPI_ERR_PENDING` to indicate that the request has not completed.

The MPI Standard does not specify what the result of the multiple completion routines is when an error occurs. For example, in an `MPI_WAITALL`, does the routine wait for all requests to either fail or complete, or does it return immediately (with the MPI definition of immediately, which means independent of actions of other MPI processes)? MPICH has chosen to make the return immediate

(alternately, local in MPI terms), and to use the error class MPI_ERR_PENDING (introduced in MPI 1.1) to indicate which requests have not completed. In most cases, only one request with an error will be detected in each call to an MPI routine that tests multiple requests. The requests that have not been processed (because an error occured in one of the requests) will have their MPI_ERROR field marked with MPI_ERR_PENDING.

Location

testsome.c

MPI_Topo_test **MPI_Topo_test**

MPI_Topo_test — Determines the type of topology (if any) associated with a communicator

Synopsis

```
int MPI_Topo_test(MPI_Comm comm, int *topo_type)
```

Input Parameter

comm communicator (handle)

Output Parameter

top_type topology type of communicator comm (choice). If the communicator has no associated topology, returns MPI_UNDEFINED.

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM

Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

See Also

MPI_Graph_create, MPI_Cart_create

Location

`topo_test.c`

MPI_Type_commit **MPI_Type_commit**

MPI_Type_commit — Commits the datatype

Synopsis

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

Input Parameter

datatype datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE

Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`type_commit.c`

MPI_Type_contiguous **MPI_Type_contiguous**

MPI_Type_contiguous — Creates a contiguous datatype

Synopsis

```
int MPI_Type_contiguous(int count,
                        MPI_Datatype old_type,
                        MPI_Datatype *new_type_p)
```

Input Parameters

count replication count (nonnegative integer)
oldtype old datatype (handle)

Output Parameter

newtype new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE

Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

`type_contiguous.c`

MPI_Type_create_darray **MPI_Type_create_darray**

MPI_Type_create_darray — create darray datatype

Synopsis

```
int MPI_Type_create_darray(int size, int rank, int ndims, int array_of_gsizes[], int array_of_distrib
```

Input Parameters

size size of process group (positive integer)
rank rank in process group (nonnegative integer)
ndims number of array dimensions as well as process grid dimensions (positive integer)
array_of_gsizes
number of elements of type oldtype in each dimension of global array (array of positive integers)
array_of_distribs
distribution of array in each dimension (array of state)

array_of_dargs
distribution argument in each dimension (array of positive integers)

array_of_psizes
size of process grid in each dimension (array of positive integers)

order array storage order flag (state)
oldtype old datatype (handle)

Output Parameter

newtype new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_create_darray.c`

MPI_Type_create_hindexed — create hindexed datatype

Synopsis

```
int MPI_Type_create_hindexed(int count,
                             int array_of_blocklengths[],
                             MPI_Aint array_of_displacements[],
                             MPI_Datatype oldtype,
                             MPI_Datatype *newtype)
```

Input Parameters

count number of blocks — also number of entries in array_of_displacements and array_of_blocklengths (integer)

array_of_blocklengths
: number of elements in each block (array of nonnegative integers)

array_of_displacements
: byte displacement of each block (array of integer)

oldtype old datatype (handle)

Output Parameter

newtype new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_create_hindexed.c`

MPI_Type_create_hvector — create hvector datatype

Synopsis

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

Input Parameters

count number of blocks (nonnegative integer)
blocklength number of elements in each block (nonnegative integer)
stride number of bytes between start of each block (integer)
oldtype old datatype (handle)

Output Parameter

newtype new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_create_hvector.c`

MPI_Type_create_indexed_block — create indexed block datatype

Synopsis

```
int MPI_Type_create_indexed_block(int count,
                                  int blocklength,
                                  int array_of_displacements[],
                                  MPI_Datatype oldtype,
                                  MPI_Datatype *newtype)
```

Input Parameters

count length of array of displacements (integer)
blocklength size of block (integer)
array_of_displacements
array of displacements (array of integer)

oldtype old datatype (handle)

Output Parameter

newtype new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_create_indexed_block.c`

MPI_Type_create_keyval — Create a attribute keyval for MPI datatypes

Synopsis

```
int MPI_Type_create_keyval(MPI_Type_copy_attr_function *type_copy_attr_fn, MPI_Type_delete_attr_funct
```

```
int *type_keyval, void *extra_state)
```

Input Parameters

type_copy_attr_fn
copy callback function for type_keyval (function)

type_delete_attr_fn
delete callback function for type_keyval (function)

extra_state extra state for callback functions

Output Parameter

type_keyval key value for future access (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`type_create_keyval.c`

MPI_Type_create_resized — create resized datatype

Synopsis

```
int MPI_Type_create_resized(MPI_Datatype oldtype,
                            MPI_Aint lb,
                            MPI_Aint extent,
                            MPI_Datatype *newtype)
```

Input Parameters

oldtype input datatype (handle)
lb new lower bound of datatype (integer)
extent new extent of datatype (integer)

Output Parameter

newtype output datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`type_create_resized.c`

MPI_Type_create_struct **MPI_Type_create_struct**

MPI_Type_create_struct — create struct datatype

Synopsis

```
int MPI_Type_create_struct(int count,
                           int array_of_blocklengths[],
                           MPI_Aint array_of_displacements[],
                           MPI_Datatype array_of_types[],
                           MPI_Datatype *newtype)
```

Input Parameters

count number of blocks (integer) — also number of entries in arrays array_of_types, array_of_displacements and array_of_blocklengths

array_of_blocklength
number of elements in each block (array of integer)

array_of_displacements
byte displacement of each block (array of integer)

array_of_types
type of elements in each block (array of handles to datatype objects)

Output Parameter

newtype new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

type_create_struct.c

MPI_Type_create_subarray **MPI_Type_create_subarray**

MPI_Type_create_subarray — create datatype subarray

Synopsis

```
int MPI_Type_create_subarray(int ndims,
                             int array_of_sizes[],
                             int array_of_subsizes[],
                             int array_of_starts[],
                             int order,
                             MPI_Datatype oldtype,
                             MPI_Datatype *newtype)
```

Input Parameters

ndims number of array dimensions (positive integer)

array_of_sizes
number of elements of type oldtype in each dimension of the full array (array of positive integers)

array_of_subsizes
number of elements of type oldtype in each dimension of the subarray (array of positive integers)

array_of_starts
starting coordinates of the subarray in each dimension (array of nonnegative integers)

order array storage order flag (state)

oldtype array element datatype (handle)

Output Parameter

newtype new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current

MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_create_subarray.c`

MPI_Type_delete_attr **MPI_Type_delete_attr**

MPI_Type_delete_attr — delete type attribute

Synopsis

```
int MPI_Type_delete_attr(MPI_Datatype type, int type_keyval)
```

Input Parameters

comm MPI datatype to which attribute is attached (handle)
type_keyval The key value of the deleted attribute (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler

MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use MPI_Error_string to get more information about this error code.

MPI_ERR_KEYVAL
Invalid keyval

Location

type_delete_attr.c

MPI_Type_dup **MPI_Type_dup**

MPI_Type_dup — duplicate a datatype

Synopsis

```
int MPI_Type_dup(MPI_Datatype datatype, MPI_Datatype *newtype)
```

Input Parameter

type datatype (handle)

Output Parameter

newtype copy of type (handle)

Notes

This is an MPI-2 function.

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler

may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`type_dup.c`

MPI_Type_extent **MPI_Type_extent**

MPI_Type_extent — Returns the extent of a datatype

Synopsis

```
int MPI_Type_extent(MPI_Datatype datatype, MPI_Aint *extent)
```

Input Parameters

datatype datatype (handle)

Output Parameter

extent datatype extent (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not*

guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`type_extent.c`

MPI_Type_free **MPI_Type_free**

MPI_Type_free — Frees the datatype

Synopsis

```
int MPI_Type_free(MPI_Datatype *datatype)
```

Input Parameter

datatype datatype that is freed (handle)

Predefined types

The MPI standard states that (in Opaque Objects)

```
MPI provides certain predefined opaque objects and predefined, static handles
to these objects. Such objects may not be destroyed.
```

Thus, it is an error to free a predefined datatype. The same section makes it clear that it is an error to free a null datatype.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler

MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see MPI_Type_commit).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., MPI_ERR_RANK).

Location

type_free.c

MPI_Type_free_keyval **MPI_Type_free_keyval**

MPI_Type_free_keyval — Free a datatype keyval

Synopsis

```
int MPI_Type_free_keyval(int *type_keyval)
```

Input Parameter

type_keyval key value (integer)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.

All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_KEYVAL
Invalid keyval

Location

`type_free_keyval.c`

MPI_Type_get_attr **MPI_Type_get_attr**

MPI_Type_get_attr — get type attribute

Synopsis

```
int MPI_Type_get_attr(MPI_Datatype type, int type_keyval, void *attribute_val, int *flag)
```

Input Parameters

type datatype to which the attribute is attached (handle)
type_keyval key value (integer)

Output Parameters

attribute_val attribute value, unless flag = false
flag false if no attribute is associated with the key (logical)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_KEYVAL
Invalid keyval

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_get_attr.c`

MPI_Type_get_extent **MPI_Type_get_extent**

MPI_Type_get_extent — get type extent

Synopsis

```
int MPI_Type_get_extent(MPI_Datatype datatype, MPI_Aint *lb, MPI_Aint *extent)
```

Input Parameter

datatype datatype to get information on (handle)

Output Parameters

lb lower bound of datatype (integer)
extent extent of datatype (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`type_get_extent.c`

MPI_Type_get_name **MPI_Type_get_name**

MPI_Type_get_name — get type name

Synopsis

```
int MPI_Type_get_name(MPI_Datatype datatype, char *type_name, int *resultlen)
```

Input Parameter

type datatype whose name is to be returned (handle)

Output Parameters

type_name the name previously stored on the datatype, or a empty string if no such name exists (string)

resultlen length of returned name (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS

No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_get_name.c`

MPI_Type_get_true_extent **MPI_Type_get_true_extent**

MPI_Type_get_true_extent — get true type extent

Synopsis

```
int MPI_Type_get_true_extent(MPI_Datatype datatype, MPI_Aint *true_lb, MPI_Aint *true_extent)
```

Input Parameter

datatype datatype to get information on (handle)

Output Parameters

true_lb true lower bound of datatype (integer)
true_extent true size of datatype (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS

No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

Location

`type_get_true_extent.c`

MPI_Type_hindexed **MPI_Type_hindexed**

MPI_Type_hindexed — Creates an indexed datatype with offsets in bytes

Synopsis

```
int MPI_Type_hindexed(int count,
                      int blocklens[],
                      MPI_Aint indices[],
                      MPI_Datatype old_type,
                      MPI_Datatype *newtype)
```

Input Parameters

count	number of blocks – also number of entries in indices and blocklens
blocklens	number of elements in each block (array of nonnegative integers)
indices	byte displacement of each block (array of MPI_Aint)
old_type	old datatype (handle)

Output Parameter

newtype	new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.
Also see the discussion for `MPI_Type_indexed` about the `indices` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler

`MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_INTERN
This error is returned when some part of the MPICH implementation is unable to acquire memory.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_hindexed.c`

MPI_Type_indexed **MPI_Type_indexed**

MPI_Type_indexed — Creates an indexed datatype

Synopsis

```
int MPI_Type_indexed(int count,
                     int blocklens[],
                     int indices[],
                     MPI_Datatype old_type,
                     MPI_Datatype *newtype)
```

Input Parameters

count number of blocks – also number of entries in indices and blocklens
blocklens number of elements in each block (array of nonnegative integers)
indices displacement of each block in multiples of old_type (array of integers)
old_type old datatype (handle)

Output Parameter

newtype new datatype (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return

value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.
The indices are displacements, and are based on a zero origin. A common error is to do something like to following

```
    integer a(100)
    integer blens(10), indices(10)
    do i=1,10
         blens(i)   = 1
10       indices(i) = 1 + (i-1)*10
    call MPI_TYPE_INDEXED(10,blens,indices,MPI_INTEGER,newtype,ierr)
    call MPI_TYPE_COMMIT(newtype,ierr)
    call MPI_SEND(a,1,newtype,...)
```

expecting this to send `a(1),a(11),...` because the indices have values `1,11,...`. Because these are *displacements* from the beginning of `a`, it actually sends `a(1+1),a(1+11),...`.
If you wish to consider the displacements as indices into a Fortran array, consider declaring the Fortran array with a zero origin

```
    integer a(0:99)
```

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

`type_indexed.c`

MPI_Type_lb **MPI_Type_lb**

MPI_Type_lb — Returns the lower-bound of a datatype

Synopsis

```
int MPI_Type_lb(MPI_Datatype datatype, MPI_Aint *displacement)
```

Input Parameters

datatype datatype (handle)

Output Parameter

displacement displacement of lower bound from origin, in bytes (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_lb.c`

MPI_Type_match_size — Find an MPI datatype matching a specified size

Synopsis

```
int MPI_Type_match_size(int typeclass, int size, MPI_Datatype *datatype)
```

Input Parameters

typeclass generic type specifier (integer)
size size, in bytes, of representation (integer)

Output Parameter

type datatype with correct type, size (handle)

Notes

`typeclass` is one of `MPI_TYPECLASS_REAL`, `MPI_TYPECLASS_INTEGER` and `MPI_TYPECLASS_COMPLEX`, corresponding to the desired typeclass. The function returns an MPI datatype matching a local variable of type `( typeclass, size )`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_match_size.c`

MPI_Type_set_attr **MPI_Type_set_attr**

MPI_Type_set_attr — set type attribute

Synopsis

```
int MPI_Type_set_attr(MPI_Datatype type, int type_keyval, void *attribute_val)
```

Input Parameters

type MPI Datatype to which attribute will be attached (handle)
keyval key value, as returned by `MPI_Type_create_keyval` (integer)
attribute_val attribute value

Notes

The type of the attribute value depends on whether C or Fortran is being used. In C, an attribute value is a pointer (`void *`); in Fortran, it is an address-sized integer.
If an attribute is already present, the delete function (specified when the corresponding keyval was created) will be called.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_KEYVAL
Invalid keyval

Location

type_set_attr.c

MPI_Type_size **MPI_Type_size**

MPI_Type_size — Return the number of bytes occupied by entries in the datatype

Synopsis

```
int MPI_Type_size(MPI_Datatype datatype, int *size)
```

Input Parameters

datatype datatype (handle)

Output Parameter

size datatype size (integer)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see MPI_Type_commit).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., MPI_ERR_RANK).

Location

type_size.c

MPI_Type_struct **MPI_Type_struct**

MPI_Type_struct — Creates a struct datatype

Synopsis

```
int MPI_Type_struct(int count,
                    int blocklens[],
                    MPI_Aint indices[],
                    MPI_Datatype old_types[],
                    MPI_Datatype *newtype)
```

Input Parameters

count	number of blocks (integer) – also number of entries in arrays array_of_types , array_of_displacements and array_of_blocklengths
blocklens	number of elements in each block (array)
indices	byte displacement of each block (array)
old_types	type of elements in each block (array of handles to datatype objects)

Output Parameter

newtype	new datatype (handle)

Notes

If an upperbound is set explicitly by using the MPI datatype `MPI_UB`, the corresponding index must be positive.
The MPI standard originally made vague statements about padding and alignment; this was intended to allow the simple definition of structures that could be sent with a count greater than one. For example,

```
struct { int a; char b; } foo;
```

may have `sizeof(foo)` > `sizeof(int) + sizeof(char)`; for example, `sizeof(foo)` == `2*sizeof(int)`. The initial version of the MPI standard defined the extent of a datatype as including an *epsilon* that would have allowed an implementation to make the extent an MPI datatype for this structure equal to `2*sizeof(int)`. However, since different systems might define different paddings, there was much discussion by the MPI Forum about what was the correct value of epsilon, and one suggestion was to define epsilon as zero. This would have been the best thing to do in MPI 1.0, particularly since the `MPI_UB` type allows the user to easily set the end of the structure. Unfortunately, this change did not make it into the final document. Currently, this routine does not add any padding, since the amount of padding needed is determined by the compiler that the user is using to build their code, not the compiler used to construct the MPI library. A later version of MPICH may provide for some natural choices of padding (e.g., multiple of the size of the largest basic member), but users are advised to never depend on this, even with vendor MPI implementations. Instead, if you define a structure datatype and wish to send or receive

multiple items, you should explicitly include an MPI_UB entry as the last member of the structure. For example, the following code can be used for the structure foo

```
blen[0] = 1; indices[0] = 0; oldtypes[0] = MPI_INT;
blen[1] = 1; indices[1] = &foo.b - &foo; oldtypes[1] = MPI_CHAR;
blen[2] = 1; indices[2] = sizeof(foo); oldtypes[2] = MPI_UB;
MPI_Type_struct( 3, blen, indices, oldtypes, &newtype );
```

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see MPI_Type_commit).

MPI_ERR_COUNT
: Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_INTERN
: This error is returned when some part of the MPICH implementation is unable to acquire memory.

Location

type_struct.c

MPI_Type_ub **MPI_Type_ub**

MPI_Type_ub — Returns the upper bound of a datatype

Synopsis

```
int MPI_Type_ub(MPI_Datatype datatype, MPI_Aint *displacement)
```

Input Parameters

datatype datatype (handle)

Output Parameter

displacement displacement of upper bound from origin, in bytes (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`type_ub.c`

MPI_Type_vector **MPI_Type_vector**

MPI_Type_vector — Creates a vector (strided) datatype

Synopsis

```
int MPI_Type_vector(int count,
                    int blocklength,
                    int stride,
                    MPI_Datatype old_type,
                    MPI_Datatype *newtype_p)
```

Input Parameters

count number of blocks (nonnegative integer)
blocklength number of elements in each block (nonnegative integer)
stride number of elements between start of each block (integer)
oldtype old datatype (handle)

Output Parameter

newtype_p new datatype (handle)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`type_vector.c`

MPI_Unpack **MPI_Unpack**

MPI_Unpack — Unpack a datatype into contiguous memory

Synopsis

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

```
MPI_Comm comm)
```

Input Parameters

inbuf input buffer start (choice)
insize size of input buffer, in bytes (integer)
position current position in bytes (integer)
outcount number of items to be unpacked (integer)
datatype datatype of each output data item (handle)
comm communicator for packed message (handle)

Output Parameter

outbuf output buffer start (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_COUNT
Invalid count argument. Count arguments must be non-negative; a count of zero is often valid.

MPI_ERR_TYPE
Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

See Also

MPI_Pack, MPI_Pack_size

Location

`unpack.c`

MPI_Unpack_external **MPI_Unpack_external**

MPI_Unpack_external — external unpack

Synopsis

```
int MPI_Unpack_external(char *datarep,
                        void *inbuf,
                        MPI_Aint insize,
                        MPI_Aint *position,
                        void *outbuf,
                        int outcount,
                        MPI_Datatype datatype)
```

Input Parameters

datarep data representation (string)
inbuf input buffer start (choice)
insize input buffer size, in bytes (integer)
outcount number of output data items (integer)
datatype datatype of output data item (handle)

Input/Output Parameter

position current position in buffer, in bytes (integer)

Output Parameter

outbuf output buffer start (choice)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_TYPE
: Invalid datatype argument. May be an uncommitted MPI_Datatype (see `MPI_Type_commit`).

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`unpack_external.c`

MPI_Unpublish_name **MPI_Unpublish_name**

MPI_Unpublish_name — Unpublish a service name published with MPI_Publish_name Input Parameters: + service_name - a service name (string) . info - implementation-specific information (handle) - port_name - a port name (string)

Synopsis

```
int MPI_Unpublish_name(char *service_name, MPI_Info info, char *port_name)
```

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler

`MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_INFO
Invalid Info

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`unpublish_name.c`

MPI_Wait **MPI_Wait**

MPI_Wait — Waits for an MPI send or receive to complete

Synopsis

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

Input Parameter

request request (handle)

Output Parameter

status status object (Status) . May be `MPI_STATUS_IGNORE`.

Note on status for send operations

For send operations, the only use of status is for `MPI_Test_cancelled` or in the case that there is an error, in which case the `MPI_ERROR` field of status will be set.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_REQUEST
: Invalid `MPI_Request`. Either null or, in the case of a `MPI_Start` or `MPI_Startall`, not a persistent request.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`wait.c`

MPI_Waitall **MPI_Waitall**

MPI_Waitall — Waits for all given communications to complete

Synopsis

```
int MPI_Waitall(int count, MPI_Request array_of_requests[], MPI_Status array_of_statuses[])
```

Input Parameters

count list length (integer)

array_of_requests
: array of request handles (array of handles)

Output Parameter

array_of_statuses
: array of status objects (array of Statuses). May be `MPI_STATUSES_IGNORE`

Notes

If one or more of the requests completes with an error, MPI_ERR_IN_STATUS is returned. An error value will be present is elements of array_of_status associated with the requests. Likewise, the MPI_ERROR field in the status elements associated with requests that have successfully completed

will be MPI_SUCCESS. Finally, those requests that have not completed will have a value of MPI_ERR_PENDING.
While it is possible to list a request handle more than once in the array_of_requests, such an action is considered erroneous and may cause the program to unexecpectedly terminate or produce incorrect results.

Note on status for send operations

For send operations, the only use of status is for `MPI_Test_cancelled` or in the case that there is an error, in which case the `MPI_ERROR` field of status will be set.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_REQUEST
: Invalid `MPI_Request`. Either null or, in the case of a `MPI_Start` or `MPI_Startall`, not a persistent request.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_IN_STATUS
: The actual error value is in the `MPI_Status` argument. This error class is returned only from the multiple-completion routines (`MPI_Testall`, `MPI_Testany`, `MPI_Testsome`, `MPI_Waitall`, `MPI_Waitany`, and `MPI_Waitsome`). The field `MPI_ERROR` in the status argument contains the error value or `MPI_SUCCESS` (no error and complete) or `MPI_ERR_PENDING` to indicate that the request has not completed.

The MPI Standard does not specify what the result of the multiple completion routines is when an error occurs. For example, in an `MPI_WAITALL`, does the routine wait for all requests to either fail or complete, or does it return immediately (with the MPI definition of immediately, which means independent of actions of other MPI processes)? MPICH has chosen to make the return immediate (alternately, local in MPI terms), and to use the error class `MPI_ERR_PENDING` (introduced in MPI 1.1) to indicate which requests have not completed. In most cases, only one request with an error will be detected in each call to an MPI routine that tests multiple requests. The requests that have

not been processed (because an error occured in one of the requests) will have their `MPI_ERROR` field marked with `MPI_ERR_PENDING`.

Location

`waitall.c`

MPI_Waitany **MPI_Waitany**

MPI_Waitany — Waits for any specified send or receive to complete

Synopsis

```
int MPI_Waitany(int count, MPI_Request array_of_requests[], int *index, MPI_Status *status)
```

Input Parameters

count list length (integer)
array_of_requests
array of requests (array of handles)

Output Parameters

index index of handle for operation that completed (integer). In the range `0` to `count-1`. In Fortran, the range is `1` to `count`.
status status object (Status). May be `MPI_STATUS_IGNORE`.

Notes

If all of the requests are `MPI_REQUEST_NULL`, then `index` is returned as `MPI_UNDEFINED`, and `status` is returned as an empty status.
While it is possible to list a request handle more than once in the array_of_requests, such an action is considered erroneous and may cause the program to unexecpectedly terminate or produce incorrect results.

Note on status for send operations

For send operations, the only use of status is for `MPI_Test_cancelled` or in the case that there is an error, in which case the `MPI_ERROR` field of status will be set.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_REQUEST
: Invalid `MPI_Request`. Either null or, in the case of a `MPI_Start` or `MPI_Startall`, not a persistent request.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`waitany.c`

MPI_Waitsome — Waits for some given communications to complete

Synopsis

```
int MPI_Waitsome(int incount, MPI_Request array_of_requests[], int *outcount, int array_of_indices[],
```

Input Parameters

incount length of array_of_requests (integer)

array_of_requests
: array of requests (array of handles)

Output Parameters

outcount number of completed requests (integer)

array_of_indices
: array of indices of operations that completed (array of integers)

array_of_statuses
: array of status objects for operations that completed (array of Status). May be `MPI_STATUSES_IGNORE`.

Notes

The array of indicies are in the range `0` to `incount - 1` for C and in the range `1` to `incount` for Fortran.
Null requests are ignored; if all requests are null, then the routine returns with `outcount` set to `MPI_UNDEFINED`.
While it is possible to list a request handle more than once in the array_of_requests, such an action is considered erroneous and may cause the program to unexecpectedly terminate or produce incorrect results.
`MPI_Waitsome` provides an interface much like the Unix `select` or `poll` calls and, in a high qualilty implementation, indicates all of the requests that have completed when `MPI_Waitsome` is called. However, `MPI_Waitsome` only guarantees that at least one request has completed; there is no guarantee that *all* completed requests will be returned, or that the entries in `array_of_indices` will be in increasing order. Also, requests that are completed while `MPI_Waitsome` is executing may or may not be returned, depending on the timing of the completion of the message.

Note on status for send operations

For send operations, the only use of status is for `MPI_Test_cancelled` or in the case that there is an error, in which case the `MPI_ERROR` field of status will be set.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_REQUEST
: Invalid `MPI_Request`. Either null or, in the case of a `MPI_Start` or `MPI_Startall`, not a persistent request.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_IN_STATUS

The actual error value is in the MPI_Status argument. This error class is returned only from the multiple-completion routines (MPI_Testall, MPI_Testany, MPI_Testsome, MPI_Waitall, MPI_Waitany, and MPI_Waitsome). The field MPI_ERROR in the status argument contains the error value or MPI_SUCCESS (no error and complete) or MPI_ERR_PENDING to indicate that the request has not completed.

The MPI Standard does not specify what the result of the multiple completion routines is when an error occurs. For example, in an MPI_WAITALL, does the routine wait for all requests to either fail or complete, or does it return immediately (with the MPI definition of immediately, which means independent of actions of other MPI processes)? MPICH has chosen to make the return immediate (alternately, local in MPI terms), and to use the error class MPI_ERR_PENDING (introduced in MPI 1.1) to indicate which requests have not completed. In most cases, only one request with an error will be detected in each call to an MPI routine that tests multiple requests. The requests that have not been processed (because an error occured in one of the requests) will have their MPI_ERROR field marked with MPI_ERR_PENDING.

Location

waitsome.c

MPI_Win_call_errhandler **MPI_Win_call_errhandler**

MPI_Win_call_errhandler — Call the error handler installed on a window object

Synopsis

```
int MPI_Win_call_errhandler(MPI_Win win, int errorcode)
```

Input Parameters

win	window with error handler (handle)
errorcode	error code (integer)

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.

All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not*

guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

Location

`win_call_errhandler.c`

MPI_Win_complete **MPI_Win_complete**

MPI_Win_complete — Completes an RMA access epoch

Synopsis

```
int MPI_Win_complete(MPI_Win win)
```

Input parameter

win window object (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_OTHER

Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_complete.c`

MPI_Win_create **MPI_Win_create**

MPI_Win_create — Create an MPI Window object for one-sided communication

Synopsis

```
int MPI_Win_create(void *base, MPI_Aint size, int disp_unit, MPI_Info info,
                   MPI_Comm comm, MPI_Win *win)
```

Input Parameters

base	initial address of window (choice)
size	size of window in bytes (nonnegative integer)
disp_unit	local unit size for displacements, in bytes (positive integer)
info	info argument (handle)
comm	communicator (handle)

Output Parameter

win	window object returned by the call (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_COMM
Invalid communicator. A common error is to use a null communicator in a call (not even allowed in `MPI_Comm_rank`).

MPI_ERR_INFO
Invalid Info

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_create.c`

MPI_Win_create_errhandler **MPI_Win_create_errhandler**

MPI_Win_create_errhandler — Create a window error handler

Synopsis

```
int MPI_Win_create_errhandler(MPI_Win_errhandler_fn *function, MPI_Errhandler *errhandler)
```

Input Parameter

function user defined error handling procedure (function)

Output Parameter

errhandler MPI error handler (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS

No error; MPI routine completed successfully.

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_create_errhandler.c`

MPI_Win_create_keyval **MPI_Win_create_keyval**

MPI_Win_create_keyval — create window keyval

Synopsis

```
int MPI_Win_create_keyval(MPI_Win_copy_attr_function *win_copy_attr_fn, MPI_Win_delete_attr_function
```

Input Parameters

win_copy_attr_fn
copy callback function for win_keyval (function)

win_delete_attr_fn
delete callback function for win_keyval (function)

extra_state extra state for callback functions

Output Parameter

win_keyval key value for future access (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_create_keyval.c`

MPI_Win_delete_attr **MPI_Win_delete_attr**

MPI_Win_delete_attr — delete window attribute

Synopsis

```
int MPI_Win_delete_attr(MPI_Win win, int win_keyval)
```

Input Parameters

win window from which the attribute is deleted (handle)
win_keyval key value (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_KEYVAL
Invalid keyval

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_delete_attr.c`

MPI_Win_fence **MPI_Win_fence**

MPI_Win_fence — Perform an MPI fence synchronization on a MPI window

Synopsis

```
int MPI_Win_fence(int assert, MPI_Win win)
```

Input Parameters

assert program assertion (integer)
win window object (handle)

Notes

The `assert` argument is used to indicate special conditions for the fence that an implementation may use to optimize the `MPI_Win_fence` operation. The value zero is always correct. Other assertion values may be or'ed together. Assertions that are valid for `MPI_Win_fence` are:

MPI_MODE_NOSTORE
the local window was not updated by local stores (or local get or receive calls) since last synchronization.

MPI_MODE_NOPUT
the local window will not be updated by put or accumulate calls after the fence call, until the ensuing (fence) synchronization.

MPI_MODE_NOPRECEDE
the fence does not complete any sequence of locally issued RMA calls. If this assertion is given by any process in the window group, then it must be given by all processes in the group.

MPI_MODE_NOSUCCEED
the fence does not start any sequence of locally issued RMA calls. If the assertion is given by any process in the window group, then it must be given by all processes in the group.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_OTHER
: Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_WIN
: Invalid MPI window object

Location

`win_fence.c`

MPI_Win_free **MPI_Win_free**

MPI_Win_free — Free an MPI RMA window

Synopsis

```
int MPI_Win_free(MPI_Win *win)
```

Input Parameter

win window object (handle)
: Notes: If successfully freed, `win` is set to `MPI_WIN_NULL`.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler

may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

Location

`win_free.c`

MPI_Win_free_keyval **MPI_Win_free_keyval**

MPI_Win_free_keyval — free window keyval

Synopsis

```
int MPI_Win_free_keyval(int *win_keyval)
```

Input Parameter

win_keyval key value (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_KEYVAL
Invalid keyval

Location

`win_free_keyval.c`

MPI_Win_get_attr **MPI_Win_get_attr**

MPI_Win_get_attr — Get attribute cached on an MPI window object

Synopsis

```
int MPI_Win_get_attr(MPI_Win win, int win_keyval, void *attribute_val, int *flag)
```

Input Parameters

win window to which the attribute is attached (handle)
win_keyval key value (integer)

Output Parameters

attribute_val attribute value, unless flag = false
flag false if no attribute is associated with the key (logical)

Notes

The following attributes are predefined for all MPI Window objects:
MPI_WIN_BASE
window base address.

MPI_WIN_SIZE
window size, in bytes.

MPI_WIN_DISP_UNIT
displacement unit associated with the window.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_WIN
: Invalid MPI window object

MPI_ERR_KEYVAL
: Invalid keyval

MPI_ERR_OTHER
: Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_get_attr.c`

MPI_Win_get_errhandler — Get the window error handler

Synopsis

```
int MPI_Win_get_errhandler(MPI_Win win, MPI_Errhandler *errhandler)
```

Input Parameter

win window (handle)

Output Parameter

errhandler error handler currently associated with window (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.

All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_WIN
: Invalid MPI window object

MPI_ERR_OTHER
: Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_get_errhandler.c`

MPI_Win_get_group — Get the MPI Group of the window object

Synopsis

```
int MPI_Win_get_group(MPI_Win win, MPI_Group *group)
```

Input Parameter

win window object (handle)

Output Parameter

group group of processes which share access to the window (handle)

Notes

The group is a duplicate of the group from the communicator used to create the MPI window, and should be freed with `MPI_Group_free` when it is no longer needed.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return

value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_WIN
: Invalid MPI window object

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

MPI_ERR_OTHER
: Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_get_group.c`

MPI_Win_get_name **MPI_Win_get_name**

MPI_Win_get_name — get window name

Synopsis

```
int MPI_Win_get_name(MPI_Win win, char *win_name, int *resultlen)
```

Input Parameter

win window whose name is to be returned (handle)

Output Parameters

win_name the name previously stored on the window, or a empty string if no such name exists (string)

resultlen length of returned name (integer)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

MPI_ERR_WIN
: Invalid MPI window object

MPI_ERR_OTHER
: Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_ARG
: Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`win_get_name.c`

MPI_Win_lock **MPI_Win_lock**

MPI_Win_lock — Begin an RMA access epoch at the target process.

Synopsis

```
int MPI_Win_lock(int lock_type, int rank, int assert, MPI_Win win)
```

Input Parameters

lock_type	either `MPI_LOCK_EXCLUSIVE` or `MPI_LOCK_SHARED` (state)
rank	rank of locked window (nonnegative integer)
assert	program assertion (integer)
win	window object (handle)

Notes

The name of this routine is misleading. In particular, this routine need not block, except when the target process is the calling process.
Implementations may restrict the use of RMA communication that is synchronized by lock calls to windows in memory allocated by `MPI_Alloc_mem`. Locks can be used portably only in such memory.
The `assert` argument is used to indicate special conditions for the fence that an implementation may use to optimize the `MPI_Win_fence` operation. The value zero is always correct. Other assertion values may be or'ed together. Assertions that are valid for `MPI_Win_fence` are:
. MPI_MODE_NOCHECK - no other process holds, or will attempt to acquire a conflicting lock, while the caller holds the window lock. This is useful when mutual exclusion is achieved by other means, but the coherence operations that may be attached to the lock and unlock calls are still required.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
: No error; MPI routine completed successfully.

Location

`win_lock.c`

MPI_Win_post **MPI_Win_post**

MPI_Win_post — Start an RMA exposure epoch

Synopsis

```
int MPI_Win_post(MPI_Group group, int assert, MPI_Win win)
```

Input parameters

group group of origin processes (handle)
assert program assertion (integer)
win window object (handle)

Notes

The `assert` argument is used to indicate special conditions for the fence that an implementation may use to optimize the `MPI_Win_post` operation. The value zero is always correct. Other assertion values may be or'ed together. Assertions that are valid for `MPI_Win_post` are:

MPI_MODE_NOCHECK
the matching calls to `MPI_WIN_START` have not yet occurred on any origin processes when the call to `MPI_WIN_POST` is made. The nocheck option can be specified by a post call if and only if it is specified by each matching start call.

MPI_MODE_NOSTORE
the local window was not updated by local stores (or local get or receive calls) since last synchronization. This may avoid the need for cache synchronization at the post call.

MPI_MODE_NOPUT
the local window will not be updated by put or accumulate calls after the post call, until the ensuing (wait) synchronization. This may avoid the need for cache synchronization at the wait call.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

Location

`win_post.c`

MPI_Win_set_attr **MPI_Win_set_attr**

MPI_Win_set_attr — set window attribute

Synopsis

```
int MPI_Win_set_attr(MPI_Win win, int win_keyval, void *attribute_val)
```

Input Parameters

win	MPI window object to which attribute will be attached (handle)
keyval	key value, as returned by `MPI_Win_create_keyval` (integer)
attribute_val	attribute value

Notes

The type of the attribute value depends on whether C or Fortran is being used. In C, an attribute value is a pointer (`void *`); in Fortran, it is an address-sized integer.
If an attribute is already present, the delete function (specified when the corresponding keyval was created) will be called.

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_KEYVAL
Invalid keyval

Location

`win_set_attr.c`

MPI_Win_set_errhandler **MPI_Win_set_errhandler**

MPI_Win_set_errhandler — Set window error handler

Synopsis

```
int MPI_Win_set_errhandler(MPI_Win win, MPI_Errhandler errhandler)
```

Input Parameters

win window (handle)
errhandler new error handler for window (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

Location

`win_set_errhandler.c`

MPI_Win_set_name **MPI_Win_set_name**

MPI_Win_set_name — set the window name

Synopsis

```
int MPI_Win_set_name(MPI_Win win, char *win_name)
```

Input Parameters

win window whose identifier is to be set (handle)
win_name the character string which is remembered as the name (string)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`win_set_name.c`

MPI_Win_start **MPI_Win_start**

MPI_Win_start — Start an RMA access epoch for MPI

Synopsis

```
int MPI_Win_start(MPI_Group group, int assert, MPI_Win win)
```

Input Parameters

group group of target processes (handle)
assert program assertion (integer)
win window object (handle)

Notes

The `assert` argument is used to indicate special conditions for the fence that an implementation may use to optimize the `MPI_Win_start` operation. The value zero is always correct. Other assertion values may be or'ed together. Assertions tha are valid for `MPI_Win_start` are:

MPI_MODE_NOCHECK
the matching calls to `MPI_WIN_POST` have already completed on all target processes when the call to `MPI_WIN_START` is made. The nocheck option can be specified in a start call if and only if it is specified in each matching post call. This is similar to the optimization of ready-send that may save a handshake when the handshake is implicit in the code. (However, ready-send is matched by a regular receive, whereas both start and post must specify the nocheck option.)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

win_start.c

MPI_Win_test **MPI_Win_test**

MPI_Win_test — Test whether an RMA exposure epoch has completed

Synopsis

```
int MPI_Win_test(MPI_Win win, int *flag)
```

Input Parameter

win window object (handle)

Output Parameter

flag success flag (logical)

Notes

This is the nonblocking version of MPI_Win_wait.

Notes for Fortran

All MPI routines in Fortran (except for MPI_WTIME and MPI_WTICK) have an additional argument ierr at the end of the argument list. ierr is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the call statement.
All MPI objects (e.g., MPI_Datatype, MPI_Comm) are of type INTEGER in Fortran.

Errors

All MPI routines (except MPI_Wtime and MPI_Wtick) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with MPI_Comm_set_errhandler (for communicators), MPI_File_set_errhandler (for files), and MPI_Win_set_errhandler (for RMA windows). The MPI-1 routine MPI_Errhandler_set may be used but its use is deprecatd. The predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_OTHER

Other error; use `MPI_Error_string` to get more information about this error code.

MPI_ERR_ARG
Invalid argument. Some argument is invalid and is not identified by a specific error class (e.g., `MPI_ERR_RANK`).

Location

`win_test.c`

MPI_Win_unlock **MPI_Win_unlock**

MPI_Win_unlock — Completes an RMA access epoch at the target process

Synopsis

```
int MPI_Win_unlock(int rank, MPI_Win win)
```

Input Parameters

rank rank of window (nonnegative integer)
win window object (handle)

Notes

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.
MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_OTHER

Other error; use `MPI_Error_string` to get more information about this error code.

Location

`win_unlock.c`

MPI_Win_wait **MPI_Win_wait**

MPI_Win_wait — Completes an RMA exposure epoch

Synopsis

```
int MPI_Win_wait(MPI_Win win)
```

Input Parameter

win window object (handle)

Notes for Fortran

All MPI routines in Fortran (except for `MPI_WTIME` and `MPI_WTICK`) have an additional argument `ierr` at the end of the argument list. `ierr` is an integer and has the same meaning as the return value of the routine in C. In Fortran, MPI routines are subroutines, and are invoked with the `call` statement.
All MPI objects (e.g., `MPI_Datatype`, `MPI_Comm`) are of type `INTEGER` in Fortran.

Errors

All MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error value; C routines as the value of the function and Fortran routines in the last argument. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. The error handler may be changed with `MPI_Comm_set_errhandler` (for communicators), `MPI_File_set_errhandler` (for files), and `MPI_Win_set_errhandler` (for RMA windows). The MPI-1 routine `MPI_Errhandler_set` may be used but its use is deprecatd. The predefined error handler `MPI_ERRORS_RETURN` may be used to cause error values to be returned. Note that MPI does *not* guarentee that an MPI program can continue past an error; however, MPI implementations will attempt to continue whenever possible.

MPI_SUCCESS
No error; MPI routine completed successfully.

MPI_ERR_WIN
Invalid MPI window object

MPI_ERR_OTHER
Other error; use `MPI_Error_string` to get more information about this error code.

Location

win_wait.c

MPI_Wtick **MPI_Wtick**

MPI_Wtick — Returns the frequency of Wtime

Synopsis

```
double MPI_Wtick( void )
```

Return value

Time in frequency of the values returned by MPI_Wtime

Notes

See Also

also: MPI_Wtime, MPI_Comm_get_attr, MPI_Attr_get

Location

wtick.c

MPI_Wtime **MPI_Wtime**

MPI_Wtime — Returns an elapsed time on the calling processor

Synopsis

```
double MPI_Wtime( void )
```

Return value

Time in seconds since an arbitrary time in the past.

Notes

This is intended to be a high-resolution, elapsed (or wall) clock. See MPI_WTICK to determine the resolution of MPI_WTIME. If the attribute MPI_WTIME_IS_GLOBAL is defined and true, then the value is synchronized across all processes in MPI_COMM_WORLD.

Notes for Fortran

This is a function, declared as `DOUBLE PRECISION MPI_WTIME()` in Fortran.

See Also

also: MPI_Wtick, MPI_Comm_get_attr, MPI_Attr_get

Location

`wtime.c`