# MPI_Barrier

Blocks until all process have reached this routine.

## Synopsis

```
#include "mpi.h"
int MPI_Barrier ( comm )
MPI_Comm comm;
```

## Input Parameters

**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.

## Algorithm

If the underlying device cannot do better, a tree-like or combine algorithm is used to broadcast a message wto all members of the communicator. We can modifiy this to use "blocks" at a later time (see `MPI_Bcast`).

## 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_Errhandler_set`; 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.

**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