**int MPI\_Init (int \*argc, char \*\*argv)**

**int MPI\_Finalize ()**

**int MPI\_Comm\_size (MPI\_Comm comm, int \*size)**

**int MPI\_Comm\_rank (MPI\_Comm comm, int \*rank)**

**int MPI\_Send (void \*buf, int count, MPI\_Datatype datatype, int dest, int tag, MPI\_Comm comm)**

**int MPI\_Recv (void \*buf, int count, MPI\_Datatype datatype, int source, int tag, MPI\_Comm comm, MPI\_Status \*status)**

**int MPI\_Sendrecv (void \*sendbuf, int sendcount, MPI\_Datatype senddatatype, int dest, int sendtag, void \*recvbuf, int recvcount, MPI\_Datatype recvdatatype, int source, int recvtag, MPI\_Comm comm, MPI\_Status \*status)**

**int MPI\_Sendrecv\_replace (void \*buff, int count, MPI\_Datatype datatype, int dest, int sendtag, int source, int recvtag, MPI\_Comm comm, MPI\_Status \*status)**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**int MPI\_Barrier (MPI\_Comm comm)**

**int MPI\_Allreduce (void \*sendbuf, void \*recvbuf, int count, MPI\_Datatype datatype, MPI\_Op op, MPI\_Comm comm)**

1. **int MPI\_Bcast (void \*buf, int count, MPI\_Datatype datatype, int source, MPI\_Comm comm)**
2. **int MPI\_Reduce (void \*sendbuf, void \*recvbuf, int count, MPI\_Datatype datatype, MPI\_Op op, int target, MPI\_Comm comm)**
3. **int MPI\_Scan (void \*sendbuf, void \*recvbuf, int count, MPI\_Datatype datatype, MPI\_Op op, MPI\_Comm comm)**
4. **int MPI\_Exscan (void \*sendbuf, void \*recvbuf, int count, MPI\_Datatype datatype, MPI\_Op op, MPI\_Comm comm)**
5. **int MPI\_Gather (void \*sendbuf, int sendcount, MPI\_Datatype senddatatype, void \*recvbuf, int recvcount, MPI\_Datatype recvdatatype, int target, MPI\_Comm comm)**
6. **int MPI\_Gatherv (void \*sendbuf, int sendcount, MPI\_Datatype senddatatype, void \*recvbuf, int \*recvcounts, int \*displs, MPI\_Datatype recvdatatype, int target, MPI\_Comm comm)**
7. **int MPI\_Allgather (void \*sendbuf, int sendcount, MPI\_Datatype senddatatype, void \*recvbuf, int recvcount, MPI\_Datatype recvdatatype, MPI\_Comm comm)**
8. **int MPI\_Allgatherv (void \*sendbuf, int sendcount, MPI\_Datatype senddatatype, void \*recvbuf, int \*recvcounts, int \*displs, MPI\_Datatype recvdatatype, MPI\_Comm comm)**
9. **int MPI\_Scatter (void \*s, int sendcount, MPI\_Datatype senddatatype, void \*recvbuf, int recvcount, MPI\_Datatype recvdatatype, int source, MPI\_Comm comm)**
10. **int MPI\_Scatterv (void \*sendbuf, int \*sendcounts, int \*displs, MPI\_Datatype senddatatype, void \*recvbuf, int recvcount, MPI\_Datatype recvdatatype, int source, MPI\_Comm comm)**

**int MPI\_Alltoall (void \*sendbuf, int sendcount, MPI\_Datatype senddatatype, void \*recvbuf, int recvcount, MPI\_Datatype recvdatatype, MPI\_Comm comm)**

**int MPI\_Comm\_split (MPI\_Comm comm, int color, int key, MPI\_Comm \*newcomm)**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**MPI\_THREAD\_SINGLE:** only one thread will execute (the same as initializing the environment with MPI\_Init ())

**MPI\_THREAD\_FUNNELED:** only the master thread can make MPI calls

**MPI\_THREAD\_SERIALIZED:** all threads can make MPI calls, but only one thread at a time can be in such state

**MPI\_THREAD\_MULTIPLE:** all threads can make simultaneous MPI calls without any constraints

-> Required is the aimed support level

-> Provided is the support level provided by the MPI implementation

**int MPI\_Init\_thread (int \*argc, char \*\*\*argv, int required, int \*provided)**