Appendix A

Basic MPI Routines

The following is a collection of MPI routines that is sufficient for most programs in the text. A very large number of routines are provided in MPI. The routines described here are divided into preliminaries (those for establishing the environment and related matters), basic point-to-point message-passing, and collective message-passing. The complete set of routines and additional details can be found in Gropp, Lusk, and Skjellum (1999), Gropp, Lusk, and Thakur (1999), and Gropp et al. (1998).

PRELIMINARIES

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

ACTIONS: Initializes MPI environment.

PARAMETERS:

`*argc`	argument from `main()`
`**argv[]`	argument from `main()`

```
int MPI_Finalize(void)
```

ACTIONS: Terminates MPI execution environment.

PARAMETERS: None.

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

ACTIONS: Determines rank of process in communicator.

PARAMETERS:

`comm`	communicator
`*rank`	rank (returned)

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

ACTIONS: Determines size of group associated with communicator.

PARAMETERS:

`comm`	communicator
`*size`	size of group (returned)

```
double MPI_Wtime(void)
```

ACTIONS: Returns elapsed time from some point in past, in seconds.

PARAMETERS: None.

POINT-TO-POINT MESSAGE-PASSING

MPI defines various datatypes for `MPI_Datatype`, mostly with corresponding C datatypes, including

```
MPI_CHAR        signed char
MPI_INT         signed int
MPI_FLOAT       float
```

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

ACTIONS: Sends message (blocking).

PARAMETERS:

`*buf`	send buffer
`count`	number of entries in buffer
`datatype`	data type of entries
`dest`	destination process rank
`tag`	message tag
`comm`	communicator

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

ACTIONS: Receives message (blocking).

PARAMETERS:

`*buf`	receive buffer (loaded)
`count`	max number of entries in buffer
`datatype`	data type of entries
`source`	source process rank
`tag`	message tag
`comm`	communicator
`*status`	status (returned)

In receive routines, `MPI_ANY_TAG` in `tag` and `MPI_ANY_SOURCE` in `source` matches with anything. The return status is a structure with at least three members:

`status -> MPI_SOURCE`	rank of source of message
`status -> MPI_TAG`	tag of source message
`Status -> MPI_ERROR`	potential errors

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

ACTIONS: Starts a nonblocking send.

PARAMETERS:

`*buf`	send buffer
`count`	number of buffer elements
`datatype`	data type of elements
`dest`	destination rank
`tag`	message tag
`comm`	communicator
`*request`	request handle (returned)

Related:

`MPI_Ibsend()`	Starts a nonblocking buffered send
`MPI_Irsend()`	Starts a nonblocking ready send
`MPI_Issend()`	Starts a nonblocking synchronous send

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

ACTIONS: Begins a nonblocking receive.

PARAMETERS:

`*buf`	receive buffer address (loaded)

`count`	number of buffer elements
`datatype`	data type of elements
`source`	source rank
`tag`	message tag
`comm`	communicator
`*request`	request handle (returned)

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

ACTIONS: Waits for an MPI send or receive to complete and then returns.

PARAMETERS:

`*request`	request handle
`*status`	status (same as return status of `MPI_recv()` if waiting for this.

Related:

`MPI_Waitall()`	Wait for all processes to complete (additional parameters)
`MPI_Waitany()`	Wait for any process to complete (additional parameters)
`MPI_Waitsome()`	Wait for some processes to complete (additional parameters)

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

ACTIONS: Tests for completion of a nonblocking operation.

PARAMETERS:

`*request`	request handle
`*flag`	true if operation completed (returned)
`*status`	status (returned)

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

ACTIONS: Blocking test for a message (without receiving message).

PARAMETERS:

`source`	source process rank
`tag`	message tag
`comm`	communicator
`*status`	status (returned)

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

ACTIONS: Nonblocking test for a message (without receiving message).

PARAMETERS:

`source`	source process rank
`tag`	message tag

`comm`	communicator
`*flag`	true if there is a message (returned)
`*status`	status (returned)

GROUP ROUTINES

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

ACTIONS: Blocks process until all processes have called it.

PARAMETERS:

`comm`	communicator

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

ACTIONS: Broadcasts message from root process to all processes in `comm` and itself.

PARAMETERS:

`*buf`	message buffer (loaded)
`count`	number of entries in buffer
`datatype`	data type of buffer
`root`	rank of root

```
int MPI_Alltoall(void *sendbuf, int sendcount, MPI_Datatype sendtype,
void *recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm)
```

ACTIONS: Sends data from all processes to all processes.

PARAMETERS:

`*sendbuf`	send buffer
`sendcount`	number of send buffer elements
`sendtype`	data type of send elements
`*recvbuf`	receive buffer (loaded)
`recvcount`	number of elements each receives
`recvtype`	data type of receive elements
`comm`	communicator

Related:

`MPI_Alltoallv()`	Sends data to all processes, with displacement

```
int MPI_Gather(void *sendbuf, int sendcount, MPI_Datatype sendtype,
void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm
```

```
comm)
```

ACTIONS: Gathers values for group of processes.

PARAMETERS:

`*sendbuf`	send buffer
`sendcount`	number of send buffer elements
`sendtype`	data type of send elements
`*recvbuf`	receive buffer (loaded)
`recvcount`	number of elements each receives
`recvtype`	data type of receive elements
`root`	rank of receiving process
`comm`	communicator

Related:

`MPI_Allgather()`	Gather values and distribute to all
`MPI_Gatherv()`	Gather values into specified locations
`MPI_Allgatherv()`	Gather values into specified locations and distribute to all

`MPI_Gatherv()` and `MPI_Allgatherv()` require additional parameter: `*displs` – array of displacements, after `recvcount`.

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

ACTIONS: Scatters a buffer from root in parts to group of processes.

PARAMETERS:

`*sendbuf`	send buffer
`sendcount`	number of elements send, each process
`sendtype`	data type of elements
`*recvbuf`	receive buffer (loaded)
`recvcount`	number of recv buffer elements
`recvtype`	type of recv elements
`root`	root process rank
`comm`	communicator

Related:

`MPI_Scatterv()`	Scatters a buffer in specified parts to group of processes.
`MPI_Reduce_scatter()`	Combines values and scatter results.

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

ACTIONS: Combines values on all processes to single value.

PARAMETERS:

`*sendbuf`	send buffer address
`*recvbuf`	receive buffer address
`count`	number of send buffer elements
`datatype`	data type of send elements
`op`	reduce operation. Several operations, including
`MPI_MAX`	Maximum
`MPI_MIN`	Minimum
`MPI_SUM`	Sum
`MPI_PROD`	Product
`root`	root process rank for result
`comm`	communicator

Related:

`MPI_Allreduce()`	Combine values to single value and return to all.

BIBLIOGRAPHY

GROPP, W., S. HUSS-LEDERMAN, A. LUMSDAINE, E. LUSK, B. NITZBERG, W. SAPHIR, AND M. SNIR (1998), *MPI—The Complete Reference*, Volume 2, *The MPI-2 Extensions*, MIT Press, Cambridge, MA.

GROPP, W., E. LUSK, AND A. SKJELLUM (1999), *Using MPI Portable Parallel Programming with the Message-Passing Interface*, 2nd edition, MIT Press, Cambridge, MA.

GROPP, W., E. LUSK, AND R. THAKUR (1999), *Using MPI-2 Advanced Features of the Message-Passing Interface*, MIT Press, Cambridge, MA.

SNIR, M., S. W. OTTO, S. HUSS-LEDERMAN, D. W. WALKER, AND J. DONGARRA (1998), *MPI—The Complete Reference*, Volume 1, *The MPI Core*, MIT Press, Cambridge, MA.