

```

1: #include "greetings.h"
2: #include <cassert>
3: #include <cstring>
4: #include <iostream>
5: #include <mpi.h> // MPI
6: #include <string>
7: using namespace std;
8:
9: // see http://www.open-mpi.org/doc/current
10: // for details on MPI functions
11:
12: void greetings(MPI_Comm const &icomm)
13: {
14:     int myrank, numprocs;
15:     MPI_Comm_rank(icomm, &myrank); // my MPI-rank
16:     MPI_Comm_size(icomm, &numprocs); // #MPI processes
17:     char *name = new char [MPI_MAX_PROCESSOR_NAME],
18:          *chbuf = new char [MPI_MAX_PROCESSOR_NAME];
19:
20:     int reslen, ierr;
21:     MPI_Get_processor_name( name, &reslen);
22:
23:     if (0==myrank) {
24:         cout << " " << myrank << " runs on " << name << endl;
25:         for (int i = 1; i < numprocs; ++i) {
26:             MPI_Status stat;
27:             stat.MPI_ERROR = 0; // M U S T be initialized!!
28:
29:             ierr = MPI_Recv(chbuf, MPI_MAX_PROCESSOR_NAME, MPI_CHAR, MPI_ANY_SOURCE,
MPI_ANY_TAG, icomm, &stat);
30:             assert(0==ierr);
31:
32:             cout << " " << stat.MPI_SOURCE << " runs on " << chbuf;
33:             int count;
34:             MPI_Get_count(&stat, MPI_CHAR, &count); // size of received data
35:             cout << " (length: " << count << " )" << endl;
36:             // stat.Get_error() // Error code
37:         }
38:     }
39:     else {
40:         int dest = 0;
41:         ierr = MPI_Send(name, strlen(name) + 1, MPI_CHAR, dest, myrank, icomm);
42:         assert(0==ierr);
43:     }
44:     delete [] chbuf;
45:     delete [] name;
46:     return;
47: }
48:
49:
50: void greetings_cpp(MPI_Comm const &icomm)
51: {
52:     int myrank, numprocs;
53:     MPI_Comm_rank(icomm, &myrank); // my MPI-rank
54:     MPI_Comm_size(icomm, &numprocs); // #MPI processes
55:     string name(MPI_MAX_PROCESSOR_NAME, '#'), // C++
56:          recvbuf(MPI_MAX_PROCESSOR_NAME, '#'); // C++: receive buffer, don't chan
ge size
57:
58:     int reslen, ierr;
59:     MPI_Get_processor_name(name.data(), &reslen);
60:     name.resize(reslen); // C++
61:
62:     if (0==myrank) {
63:         cout << " " << myrank << " runs on " << name << endl;
64:         for (int i = 1; i < numprocs; ++i) {
65:             MPI_Status stat;
66:             stat.MPI_ERROR = 0; // M U S T be initialized!!

```

```
67:
68:         ierr = MPI_Recv(recvbuf.data(), MPI_MAX_PROCESSOR_NAME, MPI_CHAR, i, MPI
_ANY_TAG, icom, &stat);
69:         assert(0==ierr);
70:
71:         int count;
72:         MPI_Get_count(&stat, MPI_CHAR, &count); // size of received data
73:         string const chbuf(recvbuf,0,count); // C++
74:         cout << " " << stat.MPI_SOURCE << " runs on " << chbuf;
75:         cout << " (length: " << count << " )" << endl;
76:         // stat.Get_error() // Error code
77:     }
78: }
79: else {
80:     int dest = 0;
81:     ierr = MPI_Send(name.data(), name.size(), MPI_CHAR, dest, myrank, icom);
82:     assert(0==ierr);
83: }
84: return;
85: }
```

```
1: //      general header for all functions in directory
2:
3: #ifndef GREETINGS_FILE
4: #define GREETINGS_FILE
5:
6: #include <mpi.h>
7:
8: /**      Each process finds out its host, sends this information
9:      to root process 0 which prints this information for each process.
10:      @param[in]      icomm      the MPI process group that is used.
11:      */
12:
13: void greetings (MPI_Comm const &icomm);
14: void greetings_cpp (MPI_Comm const &icomm);
15:
16: #endif
```

```
1: //          MPI code in C++.
2: //          See [Gropp/Lusk/Skjellum, "Using MPI", p.33/41 etc.]
3: //          and /opt/mpich/include/mpi2c++/comm.h for details
4:
5: #include "greetings.h"
6: #include <iostream>                // MPI
7: #include <mpi.h>
8: using namespace std;
9:
10: int main(int argc, char *argv[])
11: {
12:     MPI_Comm icomm = MPI_COMM_WORLD;
13:     MPI_Init(&argc, &argv);
14:     int myrank, numprocs;
15:     // numprocs = 1;    // delete this line when uncommenting the next line
16:     MPI_Comm_rank(icom, &myrank);                // my MPI-rank
17:     MPI_Comm_size(icom, &numprocs);
18:
19:     if (0==myrank) {
20:         cout << "\n There are " << numprocs << " processes running.\n \n";
21:     }
22:
23:     // greetings(icom);
24:     greetings_cpp(icom);
25:
26:     if (0==myrank) cout << endl;
27:
28:     MPI_Finalize();
29:
30:     return 0;
31: }
32:
33:
```