```
    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:
   26:         for (int i = 1; i < numprocs; ++i) {
   27:             MPI_Status stat;
   28:             stat.MPI_ERROR = 0;            // M U S T   be initialized!!
   29:
   30:             ierr = MPI_Recv(chbuf, MPI_MAX_PROCESSOR_NAME, MPI_CHAR, i, MPI_ANY_TAG,
 icomm, &stat);
   31:             assert(0==ierr);
   32:
   33:             cout << "   " << stat.MPI_SOURCE << " runs on " << chbuf;
   34:             int count;
   35:             MPI_Get_count(&stat, MPI_CHAR, &count); // size of received data
   36:             cout << " (length: " << count << " )" << endl;
   37:             //  stat.Get_error()        // Error code
   38:         }
   39:     }
   40:     else {
   41:         int dest = 0;
   42:         ierr = MPI_Send(name, strlen(name) + 1, MPI_CHAR, dest, myrank, icomm);
   43:         assert(0==ierr);
   44:     }
   45:     delete [] chbuf;
   46:     delete [] name;
   47:     return;
   48: }
   49:
   50:
   51: void greetings_cpp(MPI_Comm const &icomm)
   52: {
   53:     int myrank, numprocs;
   54:     MPI_Comm_rank(icomm, &myrank);                // my MPI-rank
   55:     MPI_Comm_size(icomm, &numprocs);              // #MPI processes
   56:     string name(MPI_MAX_PROCESSOR_NAME,'#'),      // C++
   57:            recvbuf(MPI_MAX_PROCESSOR_NAME,'#');   // C++: receive buffer, don't chan
ge size
   58:
   59:     int reslen, ierr;
   60:     MPI_Get_processor_name(name.data(), &reslen);
   61:     name.resize(reslen);                          // C++
   62:
   63:     if (0==myrank) {
   64:         cout << "   " << myrank << " runs on " << name << endl;
   65:         for (int i = 1; i < numprocs; ++i) {
   66:             MPI_Status stat;
```

```
   67:             stat.MPI_ERROR = 0;            // M U S T   be initialized!!
   68:
   69:             ierr = MPI_Recv(recvbuf.data(), MPI_MAX_PROCESSOR_NAME, MPI_CHAR, i, MPI
_ANY_TAG, icomm, &stat);
   70:             assert(0==ierr);
   71:
   72:             int count;
   73:             MPI_Get_count(&stat, MPI_CHAR, &count); // size of received data
   74:             string const chbuf(recvbuf,0,count);    // C++
   75:             cout << "   " << stat.MPI_SOURCE << " runs on " << chbuf;
   76:             cout << " (length: " << count << " )" << endl;
   77:             //  stat.Get_error()        // Error code
   78:         }
   79:     }
   80:     else {
   81:         int dest = 0;
   82:         ierr = MPI_Send(name.data(), name.size(), MPI_CHAR, dest, myrank, icomm);
   83:         assert(0==ierr);
   84:     }
   85:     return;
   86: }
```

```
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:
14:     MPI_Init(&argc, &argv);
15:     int myrank, numprocs;
16:     // numprocs = 1;    // delete this line when uncommenting the next line
17:     MPI_Comm_rank(icomm, &myrank);                          // my MPI-rank
18:     MPI_Comm_size(icomm, &numprocs);
19:
20:     if (0==myrank) {
21:         cout << "\n There are " << numprocs << " processes running.\n \n";
22:     }
23:
24:     greetings(icomm);
25:     greetings_cpp(icomm);
26:
27:     if (0==myrank) cout << endl;
28:
29:     MPI_Finalize();
30:
31:     return 0;
32: }
33:
34:
```