

```
1 //-----
2 // synchronous non-blocking ring
3 //-----
4 #if (defined __linux__ || (defined _AIX)
5     #include <sys/types.h>
6     #include <sys/stat.h>
7     #include <unistd.h>
8 #elif (defined _WIN32) || (defined _WIN64)
9     #include <conio.h>
10    #include <direct.h>
11 #endif
12
13 #include <mpi.h>
14 #include <stdio.h>
15 #include <iostream>
16 using namespace std;
17
18 int main(int argc, char** argv)
19 {
20     int i, n = 0, rank, size, dest, src, mess = 0;
21     double time;
22
23     MPI_Request request;
24     MPI_Status status;
25
26     MPI_Init(&argc, &argv);
27     MPI_Barrier(MPI_COMM_WORLD);
28     time = -MPI_Wtime();
29
30     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
31     MPI_Comm_size(MPI_COMM_WORLD, &size);
32     if (rank == 0) {
33         printf("\nThere are %d processes.\n", size);
34         mess = 333;
35         cout << "Select how many times the message " << mess << " should be shifted"
36              << " in the direction of a cycle of processes!\nIt should be n >= 0 and"
37              << " n < " << size << ".\n n = ";
38         cin >> n;
39         cout << endl;
40         fflush(stdout);
41     }
42     MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
43
44     for (i = 0; i < n; i++) {
45         dest = (rank + 1) % size;
46         // 'I' stands for immediate return
47         MPI_Issend(&mess, 1, MPI_INT, dest, 0, MPI_COMM_WORLD, &request);
48         printf("send: %d->%d\n", rank, dest);
49
50         src = (rank + size - 1) % size;
51         MPI_Recv(&mess, 1, MPI_INT, src, 0, MPI_COMM_WORLD, &status);
52         printf("receive: %d->%d\n", src, rank);
53     }
```

```
54     MPI_Wait(&request, &status);
55 }
56
57 printf("The message on processor %d: %d\n\n", rank, mess);
58 fflush(stdout);
59
60 time += MPI_Wtime();
61 if (rank == 0)
62     printf("Time = %10.6f s\n", time);
63
64 fflush(stdout);
65 MPI_Finalize();
66
67 return 0;
68 }
```