```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include<string.h>

int main(int argc, char** argv) {
  MPI_Init(NULL, NULL);
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);
  MPI_Request request1, request2;

  if (world_size < 2) {
    fprintf(stderr, "World size must be greater than 1 for %s\n", argv[0]);
    MPI_Abort(MPI_COMM_WORLD, 1);
  }

  char message[100];
  int len;
  strcpy(message, "Hello, World!");
  len = strlen(message);

  if (world_rank == 0) {
    MPI_Send(&message, len+1, MPI_CHAR, 1, 0, MPI_COMM_WORLD);
    MPI_Recv(&message, 100, MPI_CHAR, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    printf("Process 0 received string %s from process 1\n", message);
  } else if (world_rank == 1) {
    MPI_Send(&message, len+1, MPI_CHAR, 0, 1, MPI_COMM_WORLD);
    MPI_Recv(&message, 100, MPI_CHAR, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    printf("Process 1 received string %s from process 0\n", message);
  }

  MPI_Finalize();
}
```

```
> mpicc send_recv.c
> mpirun -n 2 ./a.out
```

```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include<string.h>

int main(int argc, char** argv) {
  MPI_Init(NULL, NULL);
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);
  MPI_Request request1, request2;

  if (world_size < 2) {
    fprintf(stderr, "World size must be greater than 1 for %s\n", argv[0]);
    MPI_Abort(MPI_COMM_WORLD, 1);
  }

  char message[100];
  int len;
  strcpy(message, "Hello, World!");
  len = strlen(message);

  if (world_rank == 0) {
    MPI_Isend(&message, len+1, MPI_CHAR, 1, 0, MPI_COMM_WORLD, &request1);
    MPI_Irecv(&message, 100, MPI_CHAR, 1, 0, MPI_COMM_WORLD, &request2);
    printf("Process 0 received string %s from process 1\n", message);
  } else if (world_rank == 1) {
    MPI_Isend(&message, len+1, MPI_CHAR, 0, 1, MPI_COMM_WORLD, &request1);
    MPI_Irecv(&message, 100, MPI_CHAR, 0, 1, MPI_COMM_WORLD, &request2);
    printf("Process 1 received string %s from process 0\n", message);
  }

  MPI_Finalize();
}
```

```
> mpicc send_recv.c
> mpirun -n 2 ./a.out
Process 0 received string Hello, World! from process 1
Process 1 received string Hello, World! from process 0
```

```c
#include <mpi.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
  int rank, nprocs;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &nprocs);
  MPI_Comm_size(MPI_COMM_WORLD, &rank);
  MPI_Barrier(MPI_COMM_WORLD);
  printf("Hello, World! I am %d of %d\n", nprocs, rank);
  fflush(stdout);
  MPI_Finalize();
  return 0;
}
```

```
> mpicc barrier.c
> mpirun -n 2 ./a.out
Hello, World! I am 0 of 2
Hello, World! I am 1 of 2
```

```c
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
  int rank, value;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  if (rank == 0) {
    printf("Enter a number to broadcast:\n");
    scanf("%d", &value);
  } else {
    printf("process %d: Before MPI_Bcast, value is %d\n", rank, value);
  }
  MPI_Bcast(&value, 1, MPI_INT, 0, MPI_COMM_WORLD);
  printf("process %d: After MPI_Bcast, value is %d\n", rank, value);
  MPI_Finalize();
  return 0;
}
```

```
> mpicc bcast.c
> mpirun -n 4 ./a.out
process 1: Before MPI_Bcast, value is 0
process 2: Before MPI_Bcast, value is 0
process 3: Before MPI_Bcast, value is 0
Enter a number to broadcast:
20
process 0: After MPI_Bcast, value is 20
process 2: After MPI_Bcast, value is 20
process 3: After MPI_Bcast, value is 20
process 1: After MPI_Bcast, value is 20
```