6.5.3 Deadlock

"A process is in a deadlock state if it is blocked waiting for a condition that will never become true" [3]. It is not hard to write MPI programs with calls to `MPI_Send` and `MPI_Recv` that cause processes to deadlock.

For example, consider two processes with ranks 0 and 1. Each wants to compute the average of `a` and `b`. Process 0 has an up-to-date value of `a`; process 1 has an up-to-date value of `b`. Process 0 must read `b` from 1; while process 1 must read `a` from 0. Consider this implementation:

```
float       a, b, c;
int         id;                    /* Process rank */
MPI_Status status;
...
if (id == 0) {
   MPI_Recv (&b, 1, MPI_FLOAT, 1, 0, MPI_COMM_WORLD, &status);
   MPI_Send (&a, 1, MPI_FLOAT, 1, 0, MPI_COMM_WORLD);
   c = (a + b) / 2.0;
} else if (id == 1) {
   MPI_Recv (&a, 1, MPI_FLOAT, 0, 0, MPI_COMM_WORLD, &status);
   MPI_Send (&b, 1, MPI_FLOAT, 0, 0, MPI_COMM_WORLD);
   c = (a + b) / 2.0;
}
```

Before calling `MPI_Send`, process 0 blocks inside `MPI_Recv`, waiting for the message from process 1 to arrive. In the same way, process 1 blocks inside `MPI_Recv`, waiting for the message from process 0 to arrive. The processes are deadlocked.

Okay, that error was fairly obvious (though you might be surprised at how often this kind of bug occurs in practice). Let's consider a more subtle error that also leads to deadlock.

We're solving the same problem. Processes 0 and 1 wish to exchange floating-point values. Here is the code:

```
float       a, b, c;
int         id;                    /* Process rank */
MPI_Status status;
...
if (id == 0) {
   MPI_Send (&a, 1, MPI_FLOAT, 1, 1, MPI_COMM_WORLD);
   MPI_Recv (&b, 1, MPI_FLOAT, 1, 1, MPI_COMM_WORLD, &status);
   c = (a + b) / 2.0;
} else if (id == 1) {
   MPI_Send (&b, 1, MPI_FLOAT, 0, 0, MPI_COMM_WORLD);
   MPI_Recv (&a, 1, MPI_FLOAT, 0, 0, MPI_COMM_WORLD, &status);
   c = (a + b) / 2.0;
}
```

Now both processes send the data before trying to receive the data, but they still deadlock. Can you see the mistake? Process 0 sends a message with tag 1 and tries to receive a message with tag 1. Meanwhile, process 1 sends a message with tag 0 and tries to receive a message with tag 0. Both processes will block inside `MPI_Recv`, because neither process will receive a message with the proper tag.

Another common error occurs when the sending process sends the message to the wrong destination process, or when the receiving process attempts to receive the message from the wrong source process.