

A call to `MPI_COMM_CREATE(comm, group, newcomm)` is equivalent to a call to `MPI_COMM_SPLIT(comm, color, key, newcomm)`, where all members of `group` provide `color = 0` and `key = rank` in `group`, and all processes that are not members of `group` provide `color = MPI_UNDEFINED`. The function `MPI_COMM_SPLIT` allows more general partitioning of a group into one or more subgroups with optional reordering.

The value of color must be non-negative.

Advice to users. This is an extremely powerful mechanism for dividing a single communicating group of processes into k subgroups, with k chosen implicitly by the user (by the number of colors asserted over all the processes). Each resulting communicator will be non-overlapping. Such a division could be useful for defining a hierarchy of computations, such as for multigrid, or linear algebra.

Multiple calls to `MPI_COMM_SPLIT` can be used to overcome the requirement that any call have no overlap of the resulting communicators (each process is of only one color per call). In this way, multiple overlapping communication structures can be created. Creative use of the `color` and `key` in such splitting operations is encouraged.

Note that, for a fixed color, the keys need not be unique. It is `MPI_COMM_SPLIT`'s responsibility to sort processes in ascending order according to this key, and to break ties in a consistent way. If all the keys are specified in the same way, then all the processes in a given color will have the relative rank order as they did in their parent group.

Essentially, making the key value zero for all processes of a given color means that one doesn't really care about the rank-order of the processes in the new communicator. (*End of advice to users.*)

Rationale. `color` is restricted to be non-negative, so as not to conflict with the value assigned to `MPI_UNDEFINED`. (*End of rationale.*)

The result of `MPI_COMM_SPLIT` on an intercommunicator is that those processes on the left with the same `color` as those processes on the right combine to create a new intercommunicator. The `key` argument describes the relative rank of processes on each side of the intercommunicator (see Figure 6.2). For those colors that are specified only on one side of the intercommunicator, `MPI_COMM_NULL` is returned. `MPI_COMM_NULL` is also returned to those processes that specify `MPI_UNDEFINED` as the color.

Example 6.2 (Parallel client-server model). The following client code illustrates how clients on the left side of an intercommunicator could be assigned to a single server from a pool of servers on the right side of an intercommunicator.

```
/* Client code */
MPI_Comm multiple_server_comm;
MPI_Comm single_server_comm;
int      color, rank, num_servers;

/* Create intercommunicator with clients and servers:
   multiple_server_comm */
...
```