进程组和通信器

进程组：一组进程，通过句柄进行操作
属于进程，不由进程组共享！

上下文：通信器的附属品，消息在一个给定的上下文上传递，
以确保不同通信器之间的信息不要相互干扰。

域内通信器：进程组和上下文组成
可以在其中组成拓扑连接方式

域间通信器：用于在分属于不同的进程组之间的进程进行通信。不能定义拓扑，也不能进行聚合通讯。

进程组的创建和释放

```
int MPI_Comm_group(MPI_Comm comm, MPI_Group * group);

int MPI_Group_intersection(MPI_Group group1,
                MPI_Group group2,
                MPI_Group * new_group);

int MPI_Group_difference(MPI_Group group1,
                MPI_Group group2,
                MPI_Group * new_group);

int MPI_Group_incl(MPI_Group group,
                int n, int * rank,
                MPI_Group * new_group);

int MPI_Group_excl(MPI_Group group,
                int n, int * rank,
                MPI_Group * new_group);
```

进程组的创建和释放

```
int MPI_Group_range_incl(MPI_Group group,
                int n, int ranges[][3],
                MPI_Group * new_group);

int MPI_Group_range_excl(MPI_Group group,
                int n, int ranges[][3],
                MPI_Group * new_group);

 intMPI_Group_free(MPI_Group * group);
```

进程组的操作

```
int MPI_Group_size(MPI_Group group, int * size);

int MPI_Group_translate_ranks(MPI_Group group1,
                int n, int * ranks1,
                MPI_Group group2, int * ranks2);

int MPI_Group_compare(MPI_Group group1,
                MPI_Group group2, int * result);
```

域内通信器的操作

```
int MPI_Comm_compare(MPI_Comm comm1,
                     MPI_Comm comm2, int * result);

int MPI_Comm_dup(MPI_Comm comm,
                                MPI_Comm * new_comm);

int MPI_Comm_create(MPI_Comm comm,
                                MPI_Group group,
                     MPI_Comm * new_comm);

int MPI_Comm_split(MPI_Comm comm, int color,
                     int key, MPI_Comm * new_comm);

int MPI_Comm_free(MPI_Comm * comm);
```

域内通信器的附加属性(Caching)

```
int MPI_Comm_create_keyval(
          MPI_Comm_copy_attr_function * copy_fun,
          MPI_Comm_delete_attr_function * del_fun,
          int * key, void * extra);

int MPI_Comm_set_attr(MPI_Comm comm,
          int key, void * value);

int MPI_Comm_get_attr(MPI_Comm comm,
          int key, void * value, int * found);

int MPI_Comm_free_keyval(int * key);
```

域间通信器

```
int MPI_Intercomm_create(MPI_Comm local_comm,
      int local_leader, MPI_Comm peer_comm,
      int peer_leader, int tag, MPI_Comm * new_comm);
```

建立域间通信器。

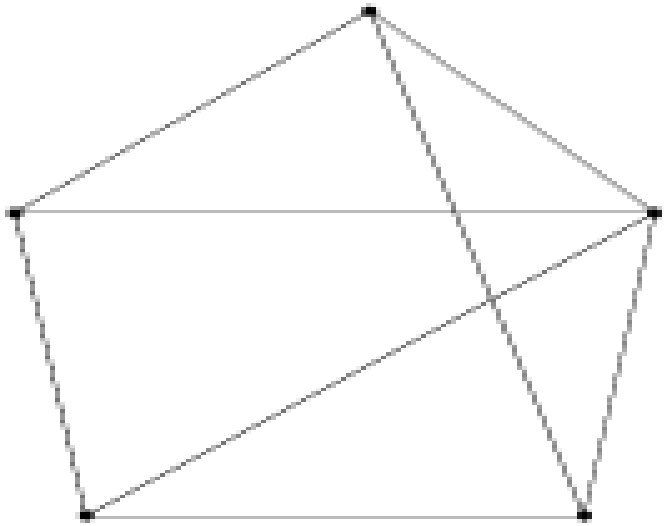
```
int MPI_Comm_merge(MPI_Comm comm, int high,
      MPI_Comm * new_comm);
```

将域间通信器转换为域内通信器。

进程的拓扑结构

这是域内通信器的附加属性，描述了进程组中各个进程之间的逻辑连接方式。可以使得程序的编制更加简洁方便也可以使得 MPI 利用硬件连接的固有属性，使得通信更加优化。

拓扑结构是一个无向图，节点是进程，边是连接方式。



笛卡儿拓扑结构

```
int MPI_Cart_create(MPI_Comm old_comm, int n_dim,
        int * dims, int * periods, int reorder,
        MPI_Comm * comm_cart);
```

创建笛卡儿拓扑结构。

```
int MPI_Dims_create(int n_nodes, int n_dim,
        int * dims);
```

维数划分。

```
int MPI_Cart_sub(MPI_Comm comm, int * flags,
        MPI_Comm * new_comm);
```

创建低维子笛卡儿结构。

笛卡儿拓扑结构

```
int MPI_Cartdim_get(MPI_Comm comm,
                    int * n_dim);
```

获得维数。

```
int MPI_Cart_get(MPI_Comm comm, int max_dim,
                 int * n_dim, int * periods,
                 int * coords);
```

获得笛卡儿拓扑结构的详细信息。

```
int MPI_Cart_rank(MPI_Comm comm, int * coords,
                  int * rank);
```

将坐标转换为秩。

```
int MPI_Cart_coords(MPI_Comm comm, int rank,
                    int max_dim, int * coords);
```

将秩转换为坐标。

笛卡儿拓扑结构

```
int MPI_Cart_shift(MPI_Comm comm, int direction,
                   int disp, int * src_rank,
                   int * dest_rank);
```

数据平移的源和目的的计算。