消息传递编程接口 MPI

（四）进程与通信器操作

为什么要进程操作

MPI提供了一些进程和通信器/通信子操作函数，通过这些操作函数，可以对进程进行分组并创建相应的通信器/通信子，灵活使用这些函数对实际编程会带来很大的方便。

目录页

Contents

进程操作函数

MPI_COMM_GROUP

MPI_COMM_GROUP(comm, group)

参数	IN comm 通信器(通信子) OUT group 进程组
C	`int MPI_Comm_group(MPI_Comm comm, MPI_Group* group)`
F77	`MPI_COMM_GROUP(COMM, GROUP, IERR)` `INTEGER COMM, GROUP, IERR`

- 创建一个通信器对应的进程组，之后就可以对该进程组进行需要的操作

MPI_GROUP_FREE

MPI_GROUP_FREE(group)

参数	INOUT group 进程组
C	int MPI_Group_free(MPI_Group* group)
F77	MPI_GROUP_FREE(GROUP, IERR) INTEGER GROUP, IERR

- 释放进程组，并返回 **MPI_GROUP_NULL**
- 当进程组被释放后，任何对该进程组的操作都是无效的

MPI_GROUP_SIZE

MPI_GROUP_SIZE(group)

参数	IN group 进程组 OUT size 进程组中进程的个数
C	`int MPI_Group_size(MPI_Group group, int * size)`
F77	`MPI_GROUP_SIZE(GROUP, SIZE, IERR)` `INTEGER GROUP, SIZE, IERR`

- 返回进程组中进程的个数
- 如果进程组是 **MPI_GROUP_EMPTY**，则返回 0

MPI_GROUP_RANK

MPI_GROUP_RANK(group)

参数	IN group 进程组 OUT rank 当前进程在进程组中的编号
C	`int MPI_Group_rank(MPI_Group group, int * rank)`
F77	`MPI_GROUP_RANK(GROUP, RANK, IERR)` `INTEGER GROUP, RANK, IERR`

- 返回当前进程在进程组中的编号
- 如果当前进程不在指定的进程组中，则返回 **MPI_UNDEFINED**

MPI_GROUP_TRANSLATE_RANKS

MPI_GROUP_TRANSLATE_RANKS(. . .)

参数	IN group1/group2 进程组1和进程组2 IN rank1 group1中有效进程编号组成的数组（全部或部分） IN n 数组ranks1和ranks2中元素的个数 OUT rank2 ranks1中编号对应的进程在group2中编号
C	int MPI_Group_translate_ranks(MPI_Group group1, int n, int * ranks1, MPI_Group group2, int * ranks2)
F77	MPI_GROUP_TRANSLATE_RANKS(GROUP1, N, RANKS1, GROUP2, RANKS2, IERR) INTEGER GROUP1, N, RANKS1(*), GROUP2, RANKS2(*), IERR

- 如果对应的进程不在 **group2** 中，则返回 **MPI_UNDEFINED**

MPI_GROUP_INCL

MPI_GROUP_INCL(group, n, ranks, newgroup)

参数	IN group 进程组 IN ranks 需要提取的所有进程的编号 IN n ranks中元素的个数和新进程组的大小 OUT newgroup 按ranks中出现的顺序新定义的进程组
C	int MPI_Group_incl(MPI_Group group, int n, int * ranks, MPI_Group * newgroup)
F77	MPI_GROUP_INCL(GROUP, N, RANKS, NEWGROUP, IERR) INTEGER GROUP, N, RANKS(*), NEWGROUP, IERR

- 从原进程组中提取部分进程组成一个新的进程组

MPI_GROUP_EXCL

MPI_GROUP_EXCL(group, n, ranks, newgroup)

参数	IN group 进程组 IN ranks 需要排除的所有进程的编号 IN n ranks中元素的个数 OUT newgroup 去掉ranks中指定的进程后组成的新进程组
C	int MPI_Group_excl(MPI_Group group, int n, int * ranks, MPI_Group * newgroup)
F77	MPI_GROUP_EXCL(GROUP, N, RANKS, NEWGROUP, IERR) INTEGER GROUP, N, RANKS(*), NEWGROUP, IERR

- 将原进程组中去除指定的部分进程后组成一个新进程组，进程顺序不变

MPI_GROUP_UNION

MPI_GROUP_UNION(group1, group2, newgroup)

参数	IN group1/group2 需要合并的两个进程组 OUT newgroup 合并后的新进程组
C	`int MPI_Group_union(MPI_Group group1, MPI_Group group2, MPI_Group * newgroup)`
F77	`MPI_GROUP_UNION(GROUP1, GROUP2, NEWGROUP, IERR)` `INTEGER GROUP1, GROUP2, NEWGROUP, IERR`

- 合并两个已有的进程组

MPI_GROUP_INTERSECTION

MPI_GROUP_INTERSECTION(group1, group2, newgroup)

参数	IN group1/group2 两个进程组 OUT newgroup 进程组1和进程组2的交集
C	int MPI_Group_intersection(MPI_Group group1, MPI_Group group2, MPI_Group * newgroup)
F77	MPI_GROUP_INTERSECTION(GROUP1, GROUP2, NEWGROUP, IERR) INTEGER GROUP1, GROUP2, NEWGROUP, IERR

- 两个进程组的交集

MPI_GROUP_DIFFERENCE

MPI_GROUP_DIFFERENCE(group1, group2, newgroup)

参数	IN group1/group2 两个进程组 OUT newgroup 进程组1和进程组2的差
C	int MPI_Group_difference(MPI_Group group1, MPI_Group group2, MPI_Group * newgroup)
F77	MPI_GROUP_DIFFERENCE(GROUP1, GROUP2, NEWGROUP, IERR) INTEGER GROUP1, GROUP2, NEWGROUP, IERR

- 在进程组1中但不在进程组2中的所有进程组成的新进程组

目录页

Contents

通信器操作函数

MPI_COMM_SIZE

MPI_COMM_SIZE(comm, size)

参数	IN comm 通信器 OUT size 通信器中进程的个数
C	`int MPI_Comm_size(MPI_Comm comm, int * size)`
F77	`MPI_COMM_SIZE(COMM, SIZE, IERR)` `INTEGER COMM, SIZE, IERR`

- 返回指定通信器中的进程个数

MPI_COMM_RANK

MPI_COMM_RANK(comm, rank)

参数	IN comm 通信器 OUT rank 通信器中进程的个数
C	`int MPI_Comm_rank(MPI_Comm comm, int * rank)`
F77	`MPI_COMM_RANK(COMM, RANK, IERR)` `INTEGER COMM, RANK, IERR`

- 返回当前进程在通信器中的进程编号

MPI_COMM_DUP

MPI_COMM_DUP(comm, newcomm)

参数	IN comm 通信器 OUT newcomm 新通信器
C	`int MPI_Comm_dup(MPI_Comm comm, MPI_Comm * newcomm)`
F77	`MPI_COMM_DUP(COMM, NEWCOMM, IERR)` `INTEGER COMM, NEWCOMM, IERR`

- 复制一个通信器

MPI_COMM_CREATE

MPI_COMM_CREATE(comm, group, newcomm)

参数	IN	comm	通信器
	IN	group	需要加入新通信器的进程
	OUT	newcomm	新通信器

C

```
int MPI_Comm_create(MPI_Comm comm, MPI_Group group,
                    MPI_Comm * newcomm)
```

F77

```
MPI_COMM_CREATE(COMM, GROUP, NEWCOMM, IERR)
INTEGER COMM, GROUP, NEWCOMM, IERR
```

- 将已有通信器中的部分进程关联到一个新的通信器中

MPI_COMM_SPLIT

MPI_COMM_SPLIT(comm, color, key, newcomm)

参数	IN comm 通信器 IN color 非负整数，用于分组，值相同的进程在同一组 IN key 非负整数，用于确定当前进程在新进程组中的顺序 OUT newcomm 新通信器
C	`int MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm * newcomm)`
F77	`MPI_COMM_SPLIT(COMM, COLOR, KEY, NEWCOMM, IERR)` `INTEGER COMM, COLOR, KEY, NEWCOMM, IERR`

- 将通信器中的所有进程分成不相交的子进程组
- 每个进程须预先设定一个 **color** 值，值相同的进程在同一个组
- 如果 **color** 的值是 **MPI_UNDEFINED**，则返回 **MPI_COMM_NULL**
- **key** 的值用来确定进程在新通信器中的顺序，若有相同值，则按原顺序

MPI_COMM_FREE

MPI_COMM_FREE(comm)

参数	INout comm 通信器
C	int MPI_Comm_free(MPI_Comm * comm)
F77	MPI_COMM_FREE(COMM, IERR) INTEGER COMM, IERR

- 释放指定的通信器，将通信器赋值 **MPI_COMM_NULL**

谢 谢

THANK YOU