Groups and Communicators

The goals of this lesson are:

- creating and using new groups
- creating and using communicators

1. What are groups and communicators ?

A group is an ordered set of processes. Let's assume a group has N processes. Each process has a unique rank inside the group. Ranks are ranging from 0 to N-1. A process can be part of multiple groups. In that case it will have a potentially different rank in any of the groups.
An MPI group is useless without attaching it to a communicator. Communicators handle the communication between processes of a group.
From a programming perspective, groups and communicators are one and the same. MPI calls make use of communicators as representing a group. However, a communicator can not exist without defining a group prior to defining the communicator.

The following steps will have to be performed when working with communicators:

1. Extract a handle of the global group – the group of the MPI_COMM_WORLD communicators
2. Define a new group, as a subset of the global group
3. Create new communicator for the newly created group
4. Determine the rank in the context of the new group
5. Perform communication within the new group using the new communicator
6. Release the communicator and group

2. Group and Communicator routines:

To extract a communicator's group:

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

Parameters:

- comm - Communicator (handle)
- group - Group in communicator (handle)

To include a set of processes into a new group:

```
int MPI_Group_incl(MPI_Group group, int n, int *ranks, MPI_Group *newgroup)
```

Parameters

- group - group (handle)
- n - number of elements in array ranks (and size of newgroup) (integer)
- ranks - ranks of processes in group to appear in newgroup (array of integers)
- newgroup - new group derived from above, in the order defined by ranks (handle)

To create a communicator for a group:

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

Parameters:

- comm - communicator (handle)
- group - group, which is a subset of the group of comm (handle)
- comm_out - new communicator (handle)

To retrieve the rank in the new group's context:

```
int MPI_Group_rank(MPI_Group group, int *rank)
```

Parameters:

- group - group (handle)
- rank - rank of the calling process in group, or MPI_UNDEFINED if the process is not a member (integer)

To mark the group and the communicator for deallocation:

```
int MPI_Comm_free(MPI_Comm *comm)
int MPI_Group_free(MPI_Group *group)
```

Parameters:

- group - group to free (handle)
- comm - Communicator to be destroyed (handle)

3. Examples

Example 1: Groups and communicators

```
#include "mpi.h"
#include <stdio.h>
#define NPROCS 8

int main(argc,argv)
int argc;
char *argv[];  {
int        rank, new_rank, sendbuf, recvbuf, numtasks,
           ranks1[4]={0,1,2,3}, ranks2[4]={4,5,6,7};
MPI_Group  orig_group, new_group;
MPI_Comm   new_comm;

MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

if (numtasks != NPROCS) {
  printf("Must specify MP_PROCS= %d. Terminating.\n",NPROCS);
  MPI_Finalize();
  exit(0);
  }

sendbuf = rank;

/* Extract the original group handle */
MPI_Comm_group(MPI_COMM_WORLD, &orig_group);

/* Divide tasks into two distinct groups based upon rank */
if (rank < NPROCS/2) {
  MPI_Group_incl(orig_group, NPROCS/2, ranks1, &new_group);
  }
else {
  MPI_Group_incl(orig_group, NPROCS/2, ranks2, &new_group);
  }

/* Create new new communicator and then perform collective communications */
MPI_Comm_create(MPI_COMM_WORLD, new_group, &new_comm);
MPI_Allreduce(&sendbuf, &recvbuf, 1, MPI_INT, MPI_SUM, new_comm);

MPI_Group_rank (new_group, &new_rank);
printf("rank= %d newrank= %d recvbuf= %d\n",rank,new_rank,recvbuf);

MPI_Finalize();
}
```

4 Exercises

1. Se da un vector cu n elemente. Sa se calculeze suma, produsul, minimul si maximul elementelor, simultan, folosind 4 grupuri de procese.