Process model and language bindings

ICHEC
Irish Centre for High-End Computing

Header files

- C

  ```
  #include <mpi.h>
  ```

- Fortran

  ```
  include 'mpif.h'
  ```

  or

  ```
  use mpi
  ```

ICHEC
Irish Centre for High-End Computing

MPI Function Format

- C:

  error = MPI_Xxxxxx(parameter, ...);

  MPI_Xxxxxx(parameter, ...);

- Fortran:

  call MPI_Xxxxxx(parameter, ..., *ierror*)

Never forget!

ICHEC
Irish Centre for High-End Computing

Initializing MPI

- C: int MPI_Init(int *argc, char ***argv)

```
#include <mpi.h>
int main(int argc, char **argv)
{
   MPI_Init(&argc, &argv);
    ....
```

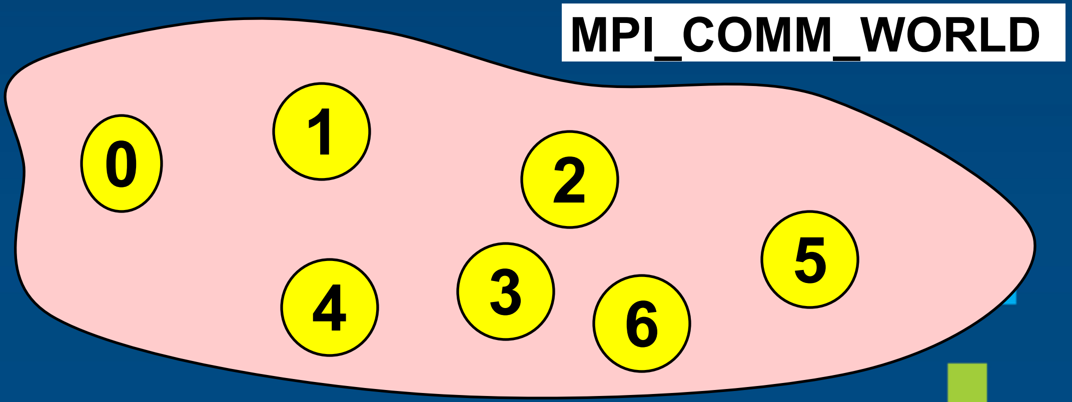
- Fortran: MPI_Init(ierror)
  integer :: ierror

```
program xxx
use mpi
implicit none
integer :: ierror
call MPI_Init(ierror)
....
```

- Must be first MPI routine that is called (except MPI_Initialized).

Communicator MPI_COMM_WORLD

- All processes of an MPI program are members of the default **communicator MPI_COMM_WORLD**.
- MPI_COMM_WORLD is a predefined **handle** in mpi.h and mpif.h.
- Each process has its own **rank** in a communicator:
  - starting with 0
  - ending with (size-1)



ICHEC
Irish Centre for High-End Computing

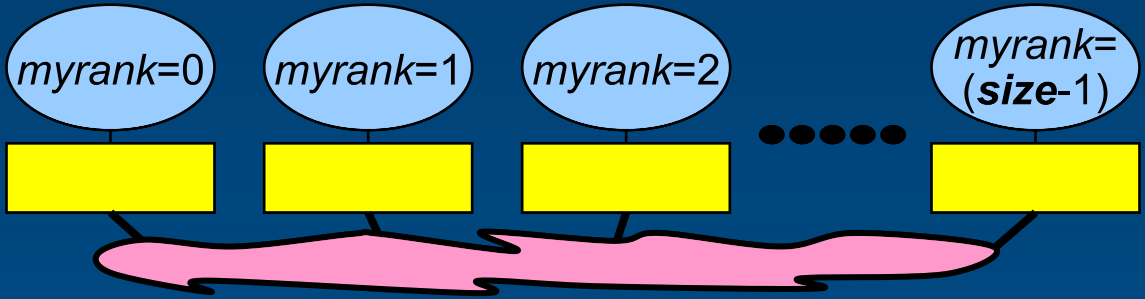
Handles

- Handles identify MPI objects.
- For the programmer, handles are
  - predefined constants in mpi.h or mpif.h
    - **example: MPI_COMM_WORLD**
    - **predefined values exist only** after MPI_Init **was called**
  - values returned by some MPI routines,
    to be stored in variables, that are defined as
    - **in Fortran: integer**
    - **in C: special MPI typedefs**
- Handles refer to internal MPI data structures

ICHEC Irish Centre for High-End Computing

Rank and Size

- C: int MPI_Comm_rank(MPI_Comm comm, int **rank*)
- Fortran: MPI_Comm_rank(comm, *rank*, *ierror*)
  integer :: comm, rank, ierror



- C: int MPI_Comm_size(MPI_Comm comm, int **size*)
- Fortran: MPI_Comm_size(comm, *size*, *ierror*)
  integer :: comm, size, ierror

ICHEC
Irish Centre for High-End Computing

Exiting MPI

- C: int MPI_Finalize()
- Fortran: MPI_Finalize(*ierror*)
  integer :: ierror

- **Must** be called last by all processes.
- After MPI_Finalize:
  - Further MPI-calls are forbidden, except MPI_Finalized.
  - Especially re-initialization with MPI_Init is forbidden

C

prog.c:

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char **argv){
    int myRank, uniSize, ierror;

    ierror=MPI_Init(&argc,&argv);
    ierror=MPI_Comm_rank(MPI_COMM_WORLD,&myRank);
    ierror=MPI_Comm_Size(MPI_COMM_WORLD,&uniSize);
    printf("I am", myRank, "of", uniSize)
    ierror=MPI_Finalize();
return 0;
}
```

Fortran

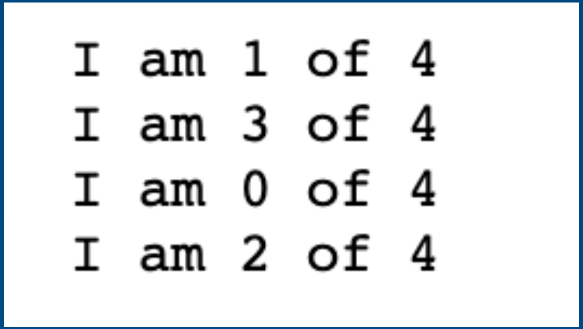
prog.f90:

```
program testMPI
use mpi
implicit none
integer :: myRank,uniSize,ierror

call MPI_Init(ierror)
call MPI_Comm_rank(MPI_COMM_WORLD,myRank,ierror)
call MPI_Comm_Size(MPI_COMM_WORLD,uniSize,ierror)
print *, ‘I am ‘, myRank, 'of ‘, uniSize
call MPI_Finalize(ierror)
end program testMPI
```

Compilation and Parallel Start

- Compilation in C: **mpicc -o** *prog prog***.c**
- Compilation in Fortran: **mpif90 -o** prog prog**.f90**
- Executing program with <u>*num*</u> processes:
  mpirun –np <u>*num*</u> ./prog



```
I am 1 of 4
I am 3 of 4
I am 0 of 4
I am 2 of 4
```

ICHEC
Irish Centre for High-End Computing

MPI Implementations

- The vendor of your computer/compiler
- MPICH
- MPI/LAM
- MPI/Pro
- openMPI
- deinoMPI