Corso di Laurea in Informatica

UNIVERSITÀ DEGLI STUDI
DI MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche, Informatiche e Matematiche

Corso
Calcolo Parallelo
Esercitazioni

Titolare del corso: prof. Luca Zanni (luca.zanni@unimore.it)

AA 2018/2019

Esercitare i concetti di base

Mpi_hello_world.c

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>int main (int argc, char* argv[]){
int rank, size;

//Inizializzo la libreria MPI
MPI_Init(&argc, &argv);

//Richiedo il numero totale di processor elements
MPI_Comm_size(MPI_COMM_WORLD, &size);

//Richiedo il mio ID tra questi          MPI_Comm_rank(MPI_COMM_WORLD,
&rank);

printf("CIAO! Sono il processo %d di %d\n", rank, size);

//Chiudo la libreria  MPI_Finalize();
return 0;
}
```

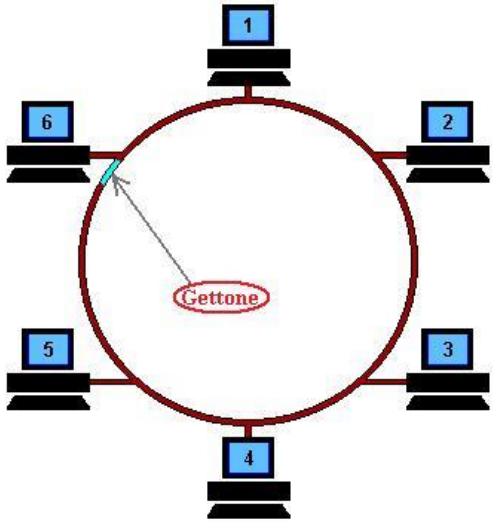
Mpi_hello_world2.c

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

int main (int argc, char* argv[]){
  int rank, size, A; MPI_Status stat;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  if (rank == 0){
    printf("Sono il processo %d, inserisci un numero: ",rank); scanf("%d",&A);
    MPI_Send(&A, 1, MPI_INT, 1, 15, MPI_COMM_WORLD);
  }
  if (rank==1){
    MPI_Recv(&A, 1, MPI_INT, 0, 15, MPI_COMM_WORLD, &stat);
    printf("Sono il processo %d, ho ricevuto il valore %d\n", rank, A);
  }

  MPI_Finalize(); return 0;
}
```

Esercizio: Mpi_ring.c

Primo esercizio con più di due processi!

Creare un programma eseguibile con un numero arbitrario di processi (n>1) in cui un valore intero è inizializzato dal processo zero e viene passato sequenzialmente ad ogni processo con ID crescente. L'esecuzione termina quando il processo 0 riceve il valore dall'ultimo processo.



MPI_Status

La funzione MPI_Recv prende come parametro anche l'indirizzo di una struttura MPI_Status (che può essere ignorata con MPI_STATUS_IGNORE).

Se passiamo un MPI_Status alla funzione esso verrà popolato con informazioni aggiuntive riguardo la ricezione del messaggio:

1.L'ID del mittente
2.Il tag del messaggio
3.La lunghezza del messaggio

Funzione per recuperare lunghezza e tipo di dati del messaggio:
MPI_Get_count(MPI_Status* status, MPI_Datatype datatype, int* count)

Esercizio: Mpi_check_status.c

Creare un programma eseguibile con due processi.
Il processo 0 invia un array con un numero random di elementi
Il processo 1 riceve il messaggio e scopre la lunghezza del messaggio mediante la funzione MPI_Get_count.

MPI_Get_count(MPI_Status* status, MPI_Datatype datatype, int* count)

Esercizio completo: Mpi_monte_carlo_pi.c

Il metodo monte-carlo per il calcolo approssimato di pi greco:

*If a circle of radius R is inscribed inside a square with side length 2R, then the area of the circle will be **pi*R^2** and the area of the square will be **(2R)^2**.*
*So the ratio of the area of the circle to the area of the square will be **pi/4**.*

*This means that, if you pick **N** points at random inside the square, approximately **N*pi/4** of those points should fall inside the circle.*

This program picks points at random inside the square. It then checks to see if the point is inside the circle (it knows it's inside the circle if x^2 + y^2 < R^2, where x and y are the coordinates of the point and R is the radius of the circle).

*The program keeps track of how many points it's picked so far (**N**) and how many of those points fell inside the circle (**M**). Pi is then approximated as follows:*

```
pi = 4*M/N
```

Esercizio completo: Mpi_monte_carlo_pi.c

- *Scrivere un programma sequenziale per il calcolo approssimato del pi greco mediante il metodo monte carlo.*

- *Parallelizzarlo mediante MPI*