# Parallel algorithms and programming
# TP 1: "Hello World"'s in MPI

Jean-Marie Madiot, Julien Herrmann
jeanmarie.madiot@ens-lyon.fr,
julien.herrmann@ens-lyon.fr

Suppose you have an MPI program written in C: `hello.c`. Instead of using `gcc`, as for standard C programs, use `mpicc` to compile MPI-based programs.

```
$ mpicc hello.c -o hello
```

Once the program is compiled, run it with `mpirun` on a number of MPI processes specified by the option `-np`:

```
$ mpirun -np 8 hello
```

## Exercise 1

Write, compile and run an MPI program named `hello1.c` such that each process prints the following line on the standard output (replacing `2` and `5` with appropriate values):

```
Hi. I am the process #2 and there are 5 of us.
```

Look up (on the Internet) the documentation for the following functions: `MPI_Init`, `MPI_Finalize`, `MPI_Comm_rank`, `MPI_Comm_size`, and how you can use them.

## Exercise 2

Write, compile and run an MPI program named `hello2.c` that uses only two processes. The first one will send the clock time to the second one, and wait for the reply. The second one will print the value received next to his own name and reply by sending the clock time. At the end, the first process prints the value received.

The following functions are relevant: `MPI_Send`, `MPI_Recv`, `MPI_Wtime`.

## Exercise 3

Extend `hello2.c` into `hello3.c` to any number of processes such that the first one sends the time to all the others, successively. All other processes respond the same way as in the previous exercise. Then the first process prints successively all the received values and the identity of the sender.

In a first version, impose the reception order. In a second version, modify the program using `MPI_ANY_SOURCE` to receive all messages in any order.

**Bonus** Try to do all the sendings before all the receptions. Does it still work? Now, try to replace `MPI_Send` with `MPI_Ssend`, does it still work? (Note: don't rely on this, for big messages `MPI_Send` could behave like `MPI_Ssend`. To do a truly non-blocking send, use `MPI_Isend`).

## Exercise 4

In `hello4.c`, the first process will send the value `1` to the next process, then waits and prints the message from the last process. Every other process will wait for a value from the previous process and send the double of the value to the next process. (The last process should send to the first one.)

What will be the result printed by the first process?

## Exercise 5

In `hello5.c` use `MPI_Reduce` to compute the factorial of the number of processes.

## Exercise 6

In `hello6.c` use smart MPI primitives in order to apply the following function to each element of the array $\{0, \ldots, n\}$ (for $n$ well above the number of processes):

$$x \mapsto x^{x \bmod 10111} \bmod 10111 \ .$$

Hint: one of the smart primitives should be `MPI_Scatter`.