Primul program în MPI

1 "Hello world!"...în MPI

Vom folosi ca exemplu o variantă a lui "Hello world!" dar care foloseşte mai multe procese care trimit mesaje "Hello" unui alt proces.

În MPI, procesele implicate în execuţia un program paralel, sunt identificate prin numere întregi. Dacă există p procese care formează un program, ele vor avea identificatorii 0, 1, ... , p-1. În următorul program fiecare proces cu identificatorul diferit de 0 va trimite un mesaj procesului 0, care va afişa mesajele primite. Astfel procesul 0 va fi un proces "master" iar celelalte vor fi procese "slave".

```
#include <stdio.h>
#include "mpi.h"        // header-ul bibliotecii MPI

main(int argc, char** argv) {
 int my_rank;       // identificatorul procesului
 int p;             // numarul de procese
 int source;        // identificatorul procesului sursa
 int dest;          // identificatorul procesului destinatie
 int tag = 50;      // eticheta mesajelor
 char message[100];    // mesajul
 MPI_Status status;    // starea unui receive

 // nici un apel MPI inainte de MPI_Init
 MPI_Init(&argc, &argv);
 // fiecare proces isi afla identificatorul
 MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
 // fiecare proces afla nr. total de procese
 MPI_Comm_size(MPI_COMM_WORLD, &p);

 if(my_rank != 0) {    // daca procesul este slave
```

```
    // construiste mesajul
    sprintf(message, "Greetings from process %d!", my_rank);
    // destinatia este procesul master
    dest = 0;

    // trimite mesajul
    MPI_Send(message, strlen(message)+1,
             MPI_CHAR, dest, tag, MPI_COMM_WORLD);
   } else {             // daca procesul este master
    for (source = 1; source < p; source++) {
      // de la fiecare proces asteapta un mesaj
      MPI_Recv(message, 100, MPI_CHAR,
               source, tag, MPI_COMM_WORLD, &status);

      // afiseaza mesajul
      printf("%s\n", message);
    }
  }

  // nici un apel MPI dupa MPI_Finalize
  MPI_Finalize();
}
```