#include <mpi.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

#include <string>

using namespace std;

MPI\_Request recv\_request[10]; maxim 10 procese

int main(int argc, char\*\* argv) {

int err = 0;

int a[10]; int total = 0; // maxim 10 procese

MPI\_Request send\_request;

err = MPI\_Init(&argc, &argv);

int world\_size;

MPI\_Comm\_size(MPI\_COMM\_WORLD, &world\_size);

int world\_rank;

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &world\_rank);

if (world\_rank != 0)

//transmitere asincrona

MPI\_Isend(&world\_rank, 1, MPI\_INT, 0, 10, MPI\_COMM\_WORLD, &send\_request);

else

{// procesul 0 asteapta de la toate cate un mesaj cu MPI\_Irecv (fara blocare)

// apoi verifica (MPI\_test) daca au venit sau nu mesajele – daca da le preia si le adauga intr-un string

string buff="hello from 0";

int flag[10];

for (int i = 0; i < world\_size; i++)

flag[i] = 0;

MPI\_Status status[10];

for (int i = 1; i < world\_size; i++)

MPI\_Irecv(a + i, 1, MPI\_INT, i, 10, MPI\_COMM\_WORLD, recv\_request + i);

while (total < world\_size - 1) {

for (int i = 1; i < world\_size; i++) {

if (flag[i] == 0) {

MPI\_Test(recv\_request + i, &flag[i], &status[i]);

if (flag[i] != 0)

{

buff += " hello from ";

buff += to\_string( a[i] );

total++;

}

}

}

}

cout << buff;

}

MPI\_Finalize();

return 0;

}