INFO-F-404 : Operating Systems II

1 Exercise

Exercise 1: In many applications a process is asking user to input a data or a command (e.g. sftp) until user enters the *exit* command.

We ask you to write a program that asks user to enter an integer value, one process has to read this value from the standard input (keyboard) and then transmit this value to other process. Each process has to output the received value and its id. Your program has to continue its execution until user enters a negative value.

Answer :

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

int main( argc, argv ) int argc; char **argv; {
int rank, value;
MPI_Init( &argc, &argv );
MPI_Comm_rank( MPI_COMM_WORLD, &rank );

do {
if (rank == 0) scanf( "%d", &value );

MPI_Bcast( &value, 1, MPI_INT, 0, MPI_COMM_WORLD );
printf( "Process %d got %d\n", rank, value );
} while (value >= 0);
MPI_Finalize( );
return 0;
}
```

Exercise 2: Write a program such described here below: process i reads an input (an integer), prints it and than passes this value to the process $i + 1$. The process $i + 1$ makes the same operations.

Answer :

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

int main( argc, argv ) int argc; char **argv; {
int rank, value, size;
MPI_Status status;
MPI_Init( &argc, &argv );
MPI_Comm_rank( MPI_COMM_WORLD, &rank );
MPI_Comm_size( MPI_COMM_WORLD, &size );

do {
if (rank == 0) {
scanf( "%d", &value );
MPI_Send( &value, 1, MPI_INT, rank + 1, 0, MPI_COMM_WORLD );
}else {
MPI_Recv( &value, 1, MPI_INT, rank - 1, 0, MPI_COMM_WORLD, &status );
if (rank < size - 1)
MPI_Send( &value, 1, MPI_INT, rank + 1, 0, MPI_COMM_WORLD );
}
printf( "Process %d got %d\n", rank, value );
} while (value >= 0);
MPI_Finalize( );
return 0;
}
```

Exercise 3: Write a program distributed among 5 processors that multiply matrix A (5×5) and a vector matrix B (5×1). Use following A and B :

$$A = \begin{pmatrix} 3 & 5 & 2 & 6 & 9 \\ 4 & 3 & 7 & 4 & 0 \\ 8 & 9 & 3 & 3 & 2 \\ 1 & 2 & 1 & 6 & 8 \\ 9 & 7 & 4 & 5 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 2 \\ 9 \\ 1 \\ 0 \\ 7 \end{pmatrix}$$

Answer :

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

int main(int *argc, char ***argv) {
int rank, value, *C;
```

```
int A[5][5] = {
{3, 5, 2, 6, 9},
{4, 3, 7, 4, 0},
{8, 9, 3, 3, 2},
{1, 2, 1, 6, 8},
{9, 7, 4, 5, 1}};
int B[5] = {2, 9, 1, 0, 7};
MPI_Init(argc, argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

if (rank == 0) C = (int*) malloc(5 * sizeof(int));
value = 0;
for (int i=0 ; i < 5 ; ++i) value += A[rank][i] * B[i];

MPI_Gather(&value, 1, MPI_INT, C, 1, MPI_INT, 0, MPI_COMM_WORLD);

if (rank == 0) {
printf("La reponse est : ");
for (int i=0 ; i < 5 ; ++i) printf("%d ", C[i]);
printf("\n");
free(C);
}
MPI_Finalize();
return 0;
}
```