Exercise 13

Write a program which decomposes an integer matrix (m x n) using a 2D MPI cartesian grid:
- handle the remainders for non multiple sizes
- fill the matrix with the row-linearized indexes

$$A_{ij} = m \cdot i + j$$

11	12	13
14	15	16
17	18	19
20	21	22

- reconstruct the absolute indexes from the local ones
- remember that in C the indexes of arrays start from 0

The program writes the matrix to file using MPI-I/O collective write and using MPI data-types:
- which data-type do you have to use?

Check the results using:
- shell Command : od -i output.dat
- parallel MPI-I/O read functions (similar to write structure)
- serial standard C and Fortran check:
  - only rank=0 performs check
  - read row-by-row in C and column-by-column in Fortran and check each element of the row/columns
  - use binary files and fread in C
  - use unformatted and access='stream' in Fortran

Which one is the most scrupolous check?
- is the Parallel MPI-I/O check enough?

	C
MPI_FILE_OPEN	`int MPI_File_open(MPI_Comm comm, char *filename, int amode, MPI_Info info, MPI_File *fh)`
MPI_FILE_DELETE	`int MPI_File_delete(char *filename, MPI_Info info)`
MPI_FILE_CLOSE	`int MPI_File_close(MPI_File *fh)`
MPI_FILE_SET_VIEW	`int MPI_File_set_view(MPI_File fh, MPI_Offset disp, MPI_Datatype etype, MPI_Datatype filetype, char *datarep, MPI_Info info)`

MPI_FILE_READ_ALL	`int MPI_File_read_all(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Status *status)`
MPI_FILE_WRITE_ALL	`int MPI_File_write_all(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Status *status)`
MPI_TYPE_CREATE_SUBARRAY	`int MPI_Type_create_subarray(int ndims, int array_of_sizes[], int array_of_subsizes[], int array_of_starts[], int order, MPI_Datatype oldtype, MPI_Datatype *newtype)`
MPI_TYPE_COMMIT	`int MPI_Type_commit(MPI_Datatype *datatype)`
MPI_TYPE_FREE	`int MPI_Type_free(MPI_Datatype *datatype)`
MPI_DIMS_CREATE	`int MPI_Dims_create(int nnodes, int ndims, int *dims)`
MPI_CART_CREATE	`int MPI_Cart_create(MPI_Comm comm_old, int ndims, int *dims, int *periods, int reorder, MPI_Comm *comm_cart)`
MPI_CART_COORDS	`int MPI_Cart_coords(MPI_Comm comm, int rank, int maxdims, int *coords)`
MPI_COMM_FREE	`int MPI_Comm_free(MPI_Comm *comm)`
MPI_INIT	`int MPI_Init(int *argc, char ***argv)`
MPI_COMM_SIZE	`int MPI_Comm_size(MPI_Comm comm, int *size)`
MPI_COMM_RANK	`int MPI_Comm_rank(MPI_Comm comm, int *rank)`
MPI_FINALIZE	`int MPI_Finalize(void)`

FORTRAN

MPI_FILE_OPEN	`MPI_FILE_OPEN(COMM, FILENAME, AMODE, INFO, FH, IERROR)`
	`CHARACTER*(*) FILENAME`
	`INTEGER COMM, AMODE, INFO, FH, IERROR`

MPI_FILE_DELETE	`MPI_FILE_DELETE(FILENAME, INFO, IERROR)`
	`CHARACTER*(*) FILENAME`
	`INTEGER INFO, IERROR`

MPI_FILE_CLOSE	`MPI_FILE_CLOSE(FH, IERROR)` `INTEGER FH, IERROR`

MPI_FILE_SET_VIEW	`MPI_FILE_SET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, INFO, IERROR)`
	`INTEGER FH, ETYPE, FILETYPE, INFO, IERROR`
	`CHARACTER*(*) DATAREP`
	`INTEGER(KIND=MPI_OFFSET_KIND) DISP`

MPI_FILE_READ_ALL	MPI_FILE_READ_ALL(FH, BUF, COUNT, DATATYPE, STATUS, IERROR) <type> BUF(*) INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
MPI_FILE_WRITE_ALL	MPI_FILE_WRITE_ALL(FH, BUF, COUNT, DATATYPE, STATUS, IERROR) <type> BUF(*) INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
MPI_TYPE_CREATE_SUBARRAY	MPI_TYPE_CREATE_SUBARRAY(NDIMS, ARRAY_OF_SIZES, ARRAY_OF_SUBSIZES, ORDER, OLDTYPE, NEWTYPE, IERROR) INTEGER NDIMS, ARRAY_OF_SIZES(*), ARRAY_OF_SUBSIZES(*), ARRAY_OF_STARTS(*), ORDER, OLDTYPE, NEWTYPE, IERROR
MPI_TYPE_COMMIT	MPI_TYPE_COMMIT(DATATYPE, IERROR) INTEGER DATATYPE, IERROR
MPI_TYPE_FREE	MPI_TYPE_FREE(DATATYPE, IERROR) INTEGER DATATYPE, IERROR
MPI_DIMS_CREATE	MPI_DIMS_CREATE(NNODES, NDIMS, DIMS, IERROR) INTEGER NNODES,NDIMS,DIMS(*),IERROR
MPI_CART_CREATE	MPI_CART_CREATE(COMM_OLD, NDIMS, PERIODS, REORDER, COMM_CART, IERROR) INTEGER COMM_OLD, NDIMS, DIMS(*), COMM_CART, IERROR LOGICAL PERIODS(*), REORDER
MPI_CART_COORDS	MPI_CART_COORDS(COMM, RANK, MAXDIMS, COORDS, IERROR) INTEGER COMM, RANK, MAXDIMS, COORDS(*), IERROR
MPI_COMM_FREE	MPI_COMM_FREE(COMM, IERROR) INTEGER COMM, IERROR
MPI_INIT	MPI_INIT(IERROR) INTEGER IERROR
MPI_COMM_SIZE	MPI_COMM_SIZE(COMM, SIZE, IERROR) INTEGER COMM, SIZE, IERRO
MPI_COMM_RANK	MPI_COMM_SIZE(COMM, SIZE, IERROR) INTEGER COMM, SIZE, IERROR
MPI_FINALIZE	MPI_FINALIZE(IERROR) INTEGER IERROR

‹ Solution 12 up Solution 13 ›