Prof. Jingke Li (lij@pdx.edu); Classes/Labs: TR 1400-1550 @FAB 150, 88-10; Office Hours: TR 1300-1400 @FAB 120-06

Lab 8: MPI File IO

1 File IO Samples

1. Read and understand the MPI program `file-in.c`. It opens a file; reads two integers from the file; and prints out their values. The input file name is provided as a command-line argument. Note that the input file is byte-encoded. To view its content, use the `od` command:

   ```
   linux> od -i data.txt
   ```

   Compile and run this program with different number of processes:

   ```
   linux> mpirun -n 4 file-in data.txt
   linux> mpirun -n 8 file-in data.txt
   ```

   Modify the buffer size and have each process read in four integers.

2. Read and understand the program `file-out.c`. This program is similar to the previous one, except that it is for writing to files. Compile and run this program with any number of processes:

   ```
   linux> mpirun -n 4 file-out output
   ```

   What do you observe? How many output files are created? What are the contents?

   Change this program so that there is only one output file, `output.all`; and have all processes write to this file. Compile and run. What do you see in this file? Can you explain?

3. Read and understand the program `file-view.c`. Pay attention to the `MPI_File_set_view()` line. Compile and run this program. Change the `offset` parameter and see the effect.

2 Array IO Program

Write an MPI program, `array-mpi.c`, to read data from a file, perform some operations, and write results to another file. The program's interface is as follows:

```
linux> mpirun -n P array-mpi <infile> <outfile>
```

The data are integers and are encoded as 32-bit (four-byte) values in the input file. (*Note:* Their values happen to be in the range `[0, 8191]`, but this should have no impact on the program.) The data size `N` is to be derived from `<infile>`'s size, which can be obtained through the following call:

```
MPI_Offset fsize;
MPI_File_get_size(fin, &fsize);
```

Here are the contents of the program:

1. Check that `N` evenly divides `P`. If not, prompt the user with a message, and gracefully terminate (i.e. make a call to `MPI_Finalize()`).

2. Allocate an array of size `N/P`, and read a section of data of the same size from the input file. You need to set a proper file-view offset for that.

3. The first operation is to compute the total sum of all data items in the input file, and have process rank 0 print out the result. Think about what local computation and what communication you need to implement. (*Hint:* You may want to take a look at `sum-mpi.c` from Lab 1; the operations are similar.)

4. The second operation is to double the value of each data item, and write the resulting data (still `N` integer values in total) to the output file.

Test your program with different `P` values on the same input, as well as on different input files.

Extra Work If you have extra time and want a challenge, you can think about removing the "`N` evenly divides `P`" constraint. Implement a generalized version in `array-mpi2.c`.

Input Data The provided program `datagen.c` can be used to generate new input data sets. It takes an integer argument, `N`, and generates `N` random integers with value in the range [0, 8191].

```
linux> ./datagen 1024 > data1k
```

A Sample Output Script

```
linux> mpirun -n 1 array-mpi in16 out
P0: psum = 69498
linux> mpirun -n 2 array-mpi in16 out
P0: psum = 34836
P1: psum = 34662
The result sum is 69498
linux> mpirun -n 4 array-mpi in16 out
P0: psum = 18846
P1: psum = 15990
P2: psum = 18253
P3: psum = 16409
The result sum is 69498
linux> od -i in16
0000000        7653        1338        3008        6847
0000020        5919        3580        6404          87
0000040        1376        4687        6916        5274
0000060        3091        4948        3203        5167
linux> od -i out
0000000       15306        2676        6016       13694
0000020       11838        7160       12808         174
0000040        2752        9374       13832       10548
0000060        6182        9896        6406       10334
0000100
```

Submission

Write a short report summarizing your work. Submit it with your `array-mpi.c` program through the "Lab 8" folder on Canvas. The submission deadline is the end of tomorrow (Friday).