CS160 – Winter 2018 – Programming Assignment #1

Due 1159pm, Jan 21, 2018

The goal of this assignment is to modify the "ring" message-passing code described in your textbook and lecture to perform a "computation". Your program will operate on any size ring of processors. Command line arguments provide a which process initializes the ring computation and which value is seeded. Your program will need to check the validity of the arguments.

You must run this program on the CSE basement lab machines (your cs160w<xy> account). Openmpi has been installed there

The computation

Given a *seed*, which is C `float`, a particular process called the *start ID* sends the *seed* to its neighbor. When that process receives it, it adds its MPI processor *id* to what it received, raises the sum to the power 1.2 and then sends the result to the next neighbor in the ring. When the *start ID* receives the result from its neighbor, it performs the same add + power computation and then sends the result to processor 0. Process 0 prints the result to `stdout`. At this point the program is complete and all MPI processors should exit.

Example, suppose the starting ID is 1, the ring is size 4 and then seed is 2.5, then full computation that is performed would be:

$$((((2.5 + 2)^{1.2} + 3)^{1.2} + 0)^{1.2} + 1)^{1.2}$$

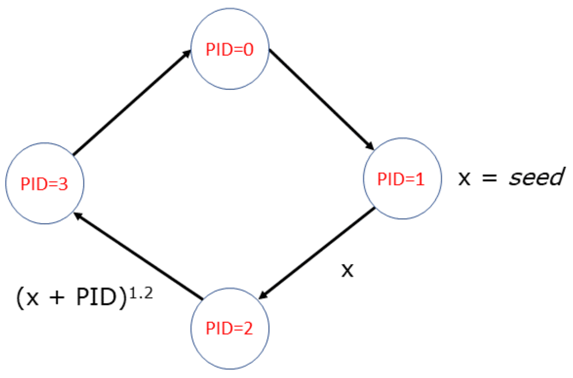

Figure-1 - Sample ring computation that begins at MPI process ID 1. Ring computation is clockwise

The program

Your program is to be call mpi-ring2 with source code mpi-ring2.c. It has the following usage

```
usage: mpi-ring2 <procid> <seedval>
```

and will be run (as an example) with four processors, seed=4.3 and starting id = 3

```
mpirun -np 4 mpi-ring2 3 4.3
```

Output

When there are no errors your program must output two lines

- status line that defines the starting MPI process ID, the seed, number of processors.
- The final result of the computation

Example:

```
$ mpirun -np 4 mpi-ring2 3 18
start 3 seed 18.000000 nproc 4
450.976562
```

Special case – number of processors = 1. If command line arguments are not in error the program should exit without any output, see examples

Error handling

Your code must handle a number of error conditions. **If any of the following errors are detected, then the usage message above should be printed on *stderr* and the program should exit with a non-zero code**. Only process id 0 should print the usage message.

Errors that must be handled (without segmentation or other faults)

1. procid passed on the command line cannot be interpreted as an int
2. seedval passed on the command line cannot be interpreted as a float
3. seedval < 0
4. the incorrect number of arguments are passed on the command line

The <procid> command line argument

- <procid> may be 0, positive or negative.
- If <procid> is 0 or positive, the messages that flow through the ring in a clockwise fashion. E.g. 1 → 2, 2→3, etc
- If <procid> is negative, the messages flow through ring in a counter clockwise fashion. E.g. 3 → 2, 2 → 1, etc
- If |<procid>| > number of processes in the ring, this is NOT an error. Instead <procid> is number of steps to go around the ring and compute which process id is the starting process for the ring computation. For example, If total number of processes is 8, the following <procid> are translated to the actual starting process id
    - 0 → 0
    - -1 -> 7
    - -2 -> 6
    - -7 -> 1
    - -8 -> 0

- -9 -> 7
- 1 -> 1
- 8 -> 0
- 9 -> 1

Handling “quoted” command line arguments

The following are all valid (non-error) invocations of your program, and should produce the identical output

- `mpirun -np 4 mpi-ring2 3 4.3`
- `mpirun -np 4 mpi-ring2 "3 " 4.3`
- `mpirun -np 4 mpi-ring2 "    3 "  "4.3"`

Validating your command line arguments

The provided files svalidate.c and svalidate.h define routines that you are free to use. They include routines to

- trim leading and trailing space from a string (char array)
- determine if a trimmed string could be converted to an `int`
- determine if a trimmed string could be converted to a `float`

The program svtest.c is a sample program that utilizes svalidate.c/svalidate.h

Files that you will turn in:

- mpi-ring2.c
- Makefile
- svalidate.c (you may turn in the version provided)
- svalidate.h (you may turn in the version provided)

Turning your program

Details provided later in the week, No later than Friday prior to the program due date.

What you will be graded on

- Correctness in both error and non-error cases
- Indentation – Your code must be indented. Choose a style and stick with it.
- Comments – You must comment your code in a reasonable way. At a minimum, your name, ucsd email address, student ID must appear in comments. All your routines must be commented with a brief description of what the routine does, what it returns and a description of the parameters. `svalidate.c` is sufficiently commented.
- `make mpi-ring2` will compile your program to the executable `mpi-ring2`
- Your `Makefile` will properly rebuild `mpi-ring2` if `svalidate.c, svalidate.h,` or `mpi-ring2.c` are updated

- Your `Makefile` has a `clean:` target that will remove intermediate object files

Example Output

```
$ mpirun -np 4 mpi-ring2 3 18
start 3 seed 18.000000 nproc 4
450.976562
$ mpirun -np 8 mpi-ring2 -2 2.54
start 6 seed 2.540000 nproc 8
27019.732422
$ mpirun -np 8 mpi-ring2 -1 2.54
start 7 seed 2.540000 nproc 8
51660.093750
$ mpirun -np 8 mpi-ring2 -0 2.54
start 0 seed 2.540000 nproc 8
4105.001465
$ mpirun -np 8 mpi-ring2 +1 2.54
start 1 seed 2.540000 nproc 8
10281.886719
$ mpirun -np 8 mpi-ring2 2 2.54
start 2 seed 2.540000 nproc 8
21171.140625
```

Special case of just one processor

```
$ mpirun -np 1 mpi-ring2 2 4.5
$
```

Some sample error conditions

```
$ mpirun -np 8 mpi-ring2 2 -2.54
usage: mpi-ring2 <procid> <seed> [debug]
-------------------------------------------------------
Primary job  terminated normally, but 1 process returned
a non-zero exit code.. Per user-direction, the job has been aborted.
-------------------------------------------------------
----------------------------------------------------------------------
----
mpirun detected that one or more processes exited with non-zero
status, thus causing
the job to be terminated. The first process to do so was:

  Process name: [[35472,1],0]
  Exit code:    255
----------------------------------------------------------------------
----
$ mpirun -np 8 mpi-ring2 2.5 2.54
usage: mpi-ring2 <procid> <seed>
-------------------------------------------------------
Primary job  terminated normally, but 1 process returned
a non-zero exit code.. Per user-direction, the job has been aborted.
-------------------------------------------------------
```

```
----------------------------------------------------------------------
----
mpirun detected that one or more processes exited with non-zero
status, thus causing
the job to be terminated. The first process to do so was:

  Process name: [[35517,1],0]
  Exit code:    255
----------------------------------------------------------------------
----
$ mpirun -np 8 mpi-ring2 2
usage: mpi-ring2 <procid> <seed>
-------------------------------------------------------
Primary job  terminated normally, but 1 process returned
a non-zero exit code.. Per user-direction, the job has been aborted.
-------------------------------------------------------
----------------------------------------------------------------------
----
mpirun detected that one or more processes exited with non-zero
status, thus causing
the job to be terminated. The first process to do so was:

  Process name: [[35529,1],5]
  Exit code:    255
```

Hints/Notes

- Build a non-MPI program to test your argument handling
- Build a non-MPI program that tests that your computation of a of starting id when given both positive and negative <procid> works properly
- Spend some time reading svalidate.c.
- Build a version of your MPI-program that prints out the intermediate computations
- Build your final program and test it under a variety of circumstances