**Exercise: MPI Greetings**

In this exercise, we’ll use the same conventions and commands as in previous exercises. You should refer back to those exercise descriptions for details on various Unix commands.

We strongly urge you to complete the exercise titled “MPI Hello World” before attempting this one.

Here are the steps for this exercise:

1. Log in to OSCER’s Linux cluster supercomputer (schooner.oscer.ou.edu).
2. Conﬁrm that you’re in your home directory:

**pwd**

/home/yourusername

1. Check that you have a SIPE subdirectory inside your home directory:

**ls**

SIPE

1. Go into your SIPE subdirectory:

**cd SIPE**

1. Conﬁrm that you’re in your SIPE subdirectory:

**pwd**

/home/yourusername/SIPE

1. See what files or subdirectories (if any) are in the current working directory:

**ls**

1. Copy the subdirectory named Greetings from Henry’s SIPE directory into your SIPE directory:

**cp -r ~hneeman/SIPE/Greetings ~/SIPE/**

1. Go into your Greetings subdirectory:

**cd Greetings**

1. Conﬁrm that you’re in your Greetings subdirectory:

**pwd**

/home/yourusername/SIPE/Greetings

1. See what files or subdirectories (if any) are in the current working directory:

**ls**

1. Choose which language you want to use (C or Fortran90), and cd into the appropriate directory:

**cd C/**

OR:

**cd Fortran90/**

1. Conﬁrm that you’re in your C or Fortran90 subdirectory:

**pwd**

/home/yourusername/SIPE/Greetings/C

OR the output of the pwd command might be:

/home/yourusername/SIPE/Greetings/Fortran90

1. See what files or subdirectories (if any) are in the current working directory:

**ls**

1. Edit the batch script greetings.sbatch to use your username and e-mail address.
2. If you haven’t already examined greetings.c (or greetings.f90), do so now.
3. Compile using the *shell script* make\_cmd:

**make\_cmd**

**NOTE**: A *shell script* is a file containing a sequence of Unix commands, which are executed like a program.

If that doesn’t work, try this:

**./make\_cmd**

That is, put a dot (period) and a slash before make\_cmd, with no blank spaces.

1. Submit the batch script file greetings.sbatch to the batch scheduler:

**sbatch bsub <greetings.sbatch**

**NOTICE** the less than symbol<which is **EXTREMELY IMPORTANT**.

You should get back output something like this:

Submitted batch job #######

Job <######> is submitted to queue <normal>.

where ####### is replaced by the batch job ID for the batch job that you’ve just submitted, and you may have a different queue name.

1. Check the status of your batch job:

**bjobs**

**squeue -u yourusername**

You’ll get one of the following outputs, either:

JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)

####### normalsipe greeting yourusername PD 0:00 2 (None)

No unfinished job found

(if you get this right after the bjobs command, try it several more times, because sometimes there’s a pause just before the batch job starts showing up, as below),

OR something like this:

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

4081250 yourusername PEND normal boomer1 greetings May 18 10:01

where ####### is replaced by a batch job ID number, and yourusername is replaced by your user name, and where PD is short for “pending,” meaning that your job is waiting to start,

OR something like this:

JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)

####### normalsipe greeting yourusername R 1:06 2 c[026-027]

JOBID USER STAT QUEUE FROM\_HOST EXEC\_HOST JOB\_NAME SUBMIT\_TIME

4081250 yourusername RUN normal boomer1 c127 greetings May 18 10:01

1. You may need to check the status of your batch job repeatedly, using the squeuebjobs command, until it runs to completion. **This may take several minutes (occasionally much longer).**

You’ll know that the batch job has finished when it no longer appears in the list of your batch jobs:

No unfinished job found

1. If you find that your batch job waits in the batch queue for more than a few minutes, you can edit your batch script to use the batch queue (partition) named debug.

Specifically, in your batch script, replace this:

#SBATCH –-partition=normal

with this:

#SBATCH –-partition=debug

1. Once your job has finished running, ﬁnd the *standard output* and *standard error* files from your job:

**ls -ltr**

Using this command, you should see files named

greetings\_#######\_stdout.txt

and

greetings\_#######\_stderr.txt

(where ####### is replaced by the batch job ID).

These files should contain the output of greetings. Ideally, the stderr file should have length zero, but it may contain a modest amount of warning messages.

1. Look at the contents of the standard output file:

**cat greetings\_#######\_stdout.txt**

(where ####### is replaced by the batch job ID).

You may want to look at the stderr file as well:

**cat greetings\_#######\_stderr.txt**

1. Is the output what you expected? Why or why not?
2. If this run had **ANY** problems, then send e-mail to:

[support@oscer.ou.edu](mailto:support@oscer.ou.edu)

which reaches all OSCER staff (including Henry), and attach the following files:

make\_cmd

makefile

greetings.c

greetings.sbatch

greetings\_#######\_stdout.txt

greetings\_#######\_stderr.txt

1. Edit the batch script greetings.sbatch to change the number of MPI processes to run, by changing the value in the following batch directive:

#SBATCH --ntasks=24

#BSUB -n 32

Try any number of processes from 1 to 40. For values less than 20, be sure to change **BOTH** the value in the batch directive above and also the value in the batch directive below, which should match the one above (if the value for the above is less than 20):

#SBATCH --ntasks-per-node=20

#BSUB -R "span[ptile=16]"

**NOTES**:

* If the value for the first batch directive above is 20 or greater, then the value for the second batch directive above should be 20.
* If the value for the first batch directive above is less than 24, then the value for the second batch directive above should be the same as the value for first batch directive above.

1. Repeat steps 17 – 23. (You won’t need to repeat step 16 yet.)
2. Edit the source file, either greetings.c or greetings.f90, as follows:

In the call to MPI\_Recv, replace this:

source

with this:

MPI\_ANY\_SOURCE

(Note that this is in all upper case letters, with underscores between the words.)

1. Repeat step 16, above, and then steps 17 – 23, above.
2. What difference(s) do you see between the output for the original version compared to the new version? How do you explain the difference(s)?
3. Repeat steps 24 – 25.
4. If you haven’t already, test this code running on 1 process. How do you explain the output?

In particular, why doesn’t process 0 produce a greeting?