**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.

Here are the steps for this exercise:

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

**pwd**

/home/yourusername

1. Check that you have a PPCC2012 subdirectory inside your home directory:

**ls**

PPCC2012

1. Copy the Greetings directory into your PPCC2012 directory:

**cp -r ~hneeman/PPCC2012/Greetings/ ~/PPCC2012/**

1. Go into your PPCC2012 subdirectory:

**cd PPCC2012**

1. Conﬁrm that you’re in your PPCC2012 subdirectory:

**pwd**

/home/yourusername/PPCC2012

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

**ls**

1. Go into your Greetings subdirectory:

**cd Greetings**

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

**pwd**

/home/yourusername/PPCC2012/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/PPCC2012/Greetings/C

OR the output of the pwd command might be:

/home/yourusername/PPCC2012/Greetings/Fortran90

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

**ls**

1. Edit the batch script greetings.bsub 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.bsub to the batch scheduler:

**bsub < greetings.bsub**

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

You should get back output something like this:

Job <######> is submitted to queue <ppcc\_q>.

where ###### is replaced by the batch job ID for the batch job that you’ve just submitted.

1. Check the status of your batch job:

**bjobs**

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

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 ppcc\_q boomer1 greetings Oct 17 14:58

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

OR something like this:

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

4081250 yourusername RUN ppcc\_q boomer1 c127 greetings Oct 17 14:58

1. You may need to check the status of your batch job repeatedly, using the bjobs 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. 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.

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 operations staff plus Henry, and attach the following files:

make\_cmd

makefile

greetings.c

greetings.bsub

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

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

1. 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 steps 16 – 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. Edit the batch script greetings.bsub to change the number of MPI processes to run. Try any number from 8 to 32.
4. Repeat steps 17 – 23. (You won’t need to repeat step 16.)
5. Why doesn’t process 0 produce a greeting?