**Exercise: MPI Greetings**

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

You’ll be running an MPI code, similar to the MPI version of the classic Hello World program.

Here are the steps for this exercise:

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

% **pwd**

/home/yourusername

1. Check that you have a NCSIPARI2011\_exercises subdirectory inside your home directory:

% **ls**

NCSIPARI2011\_exercises

1. Copy the Greetings directory into your NCSIPARI2011\_exercises directory:

% **cp -r ~hneeman/NCSIPARI2011\_exercises/Greetings/ ~/NCSIPARI2011\_exercises/**

1. Go into your NCSIPARI2011\_exercises subdirectory:

% **cd NCSIPARI2011\_exercises**

1. Conﬁrm that you’re in your NCSIPARI2011\_exercises subdirectory:

% **pwd**

/home/yourusername/NCSIPARI2011\_exercises

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 NCSIPARI2011\_exercises subdirectory:

% **pwd**

/home/yourusername/NCSIPARI2011\_exercises/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. 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.

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 <pari\_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:

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

4081250 yourusername PEND pari\_q sooner1 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:

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

4081250 yourusername RUN pari\_q sooner1 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\_######\_stdout.txt**

1. If this run had **ANY** problems, then send e-mail to:

support@oscer.ou.edu

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

make\_cmd

makefile

greetings.c

greetings.bsub

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

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

1. Edit greetings.c to change one of the arguments of the call to MPI\_Recv, replacing source with MPI\_ANY\_SOURCE.
2. Repeat steps 14 - 20.

What difference(s), if any, do you observe in the behavior of this new version of the code, compared to the original version?

1. Which rank doesn’t output a greeting? Why not?