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

   `ls`
   `PPCC2012`
4. Copy the `Greetings` directory into your `PPCC2012` directory:

   `cp -r ~hneeman/PPCC2012/Greetings/ ~/PPCC2012/`
5. Go into your `PPCC2012` subdirectory:

   `cd PPCC2012`
6. Confirm that you're in your `PPCC2012` subdirectory:

   `pwd`
   `/home/yourusername/PPCC2012`
7. See what files or subdirectories (if any) are in the current working directory:

   `ls`
8. Go into your `Greetings` subdirectory:

   `cd Greetings`
9. Confirm that you're in your `Greetings` subdirectory:

   `pwd`
   `/home/yourusername/PPCC2012/Greetings`
10. See what files or subdirectories (if any) are in the current working directory:

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

    `cd C/`

    OR:

    `cd Fortran90/`
12. 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`

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

    `ls`

14. Edit the batch script `greetings.bsub` to use your username and e-mail address.

15. If you haven't already examined `greetings.c` (or `greetings.f90`), do so now.

16. 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.

17. 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.

18. 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
    ```

19. 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
    ```

20. 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.

21. 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`**

22. Is the output what you expected? Why or why not?

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

    `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
    ```

24. 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.)

25. Repeat steps 16 – 23, above.

26. 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)?

27. Edit the batch script `greetings.bsub` to change the number of MPI processes to run. Try any number from 8 to 32.

28. Repeat steps 17 – 23. (You won't need to repeat step 16.)

29. Why doesn't process 0 produce a greeting?