Exercise: MPI Hello World

In this exercise, we'll use the same conventions and commands as in the batch computing exercise. You should refer back to the batch computing exercise description for details on various Unix commands.

You'll be running your first MPI code, specifically an 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
   ```

3. Check that you have a `NCSIPARII2011_exercises` subdirectory inside your home directory:

   ```
   ls
   NCSIPARII2011_exercises
   ```

4. Copy the `HelloWorld` directory into your `NCSIPARII2011_exercises` directory:

   ```
   cp -r ~hneeman/NCSIPARII2011_exercises/HelloWorld/ ~/NCSIPARII2011_exercises/
   ```

5. Go into your `NCSIPARII2011_exercises` subdirectory:

   ```
   cd NCSIPARII2011_exercises
   ```

6. Confirm that you're in your `NCSIPARII2011_exercises` subdirectory:

   ```
   pwd
   /home/yourusername/NCSIPARII2011_exercises
   ```

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

   ```
   ls
   ```

8. Go into your `HelloWorld` subdirectory:

   ```
   cd HelloWorld
   ```

9. Confirm that you're in your `NCSIPARII2011_exercises` subdirectory:

   ```
   pwd
   /home/yourusername/NCSIPARII2011_exercises/HelloWorld
   ```

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. Edit the batch script `hello_world.bsub` to use your username and e-mail address.
13. If you haven't already examined `hello_world.c` (or `hello_world.f90`), do so now.

14. 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.

15. Submit the batch script file `hello_world.bsub` to the batch scheduler:

    `bsub < hello_world.bsub`

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

    You should get back output something like this:

    ```
    Job <######> is submitted to queue <parii_q>.
    ```

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

16. 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  parii_q  sooner1               hello_world  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   parii_q  sooner1    c127       hello_world  Oct 17 14:58
    ```

17. 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
    ```

18. 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

    ```
    hello_world_######_stdout.txt
    ```

    and

    ```
    hello_world_######_stderr.txt
    ```

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

    These files should contain the output of `hello_world`. Ideally, the `stderr` file should have length zero.

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

    `cat hello_world_######_stdout.txt`

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

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

    `cat hello_world_######_stdout.txt`

20. 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
    hello_world.c
    hello_world.bsub
    hello_world_######_stdout.txt
    hello_world_######_stderr.txt
    ```