ME 759
High Performance for Engineering Applications
Assignment 1
Due Thursday 2/4/2020 at 9:00 PM

Submit responses to all tasks which don't specify a file name to Canvas in a file called assignment1.{txt, docx, pdf, rtf, odt} (choose one of these formats). Submit all plots (if any) on Canvas. Do not zip your Canvas submission.

All *source files* should be submitted in the `HW01` subdirectory on the `master` branch of your `git` repo. Please use the name `HW01` exactly as shown here (both in terms of capitalization & name). Other names like `hw1`, `hw01`, `HW1` will not be recognized by the repo scripts. The `HW01` subdirectory should have no subdirectories. For this assignment, your `HW01` folder should contain `task4.sh` and `task6.cpp`.

All commands or code must work on *Euler* with no modules loaded unless specified otherwise (post a question on Piazza if the term *module* is confusing; or search the ME459 slides for the term). They may behave differently on your computer, so be sure to test on *Euler* before you submit.

Please submit clean code. Consider using a formatter like clang-format.

---

1. (a) Read the files `timing.md` and `slurm_usage.md` from the `2021Spring/Assignments/general` directory of the ME759 Resource Repo. These documents set out expectations for your assignments throughout the semester.

   (b) Read the `hw_repos.md` file in the same directory and follow the instructions to create an account. This is very important and must be done in order for you to turn in all the assignments for ME759. Please create an account before the end of this week (1/29 at 9pm) so that the TAs can create your HW repository under your account. Please do not create the HW repo by yourself.

   (c) At least skim `workflow.md`. This contains a quick guide for effectively working between your local computer and *Euler*. This was also discussed in the first lecture.

2. Write one line of *bash* code for each of the following sub-tasks (assume that all the files and directories mentioned exist). The purpose of this task is to get you familiar a bit with the Linux command line. To that end, your commands should be as issued on *Euler*.

   a) Change the current directory to a subdirectory called `somedir`

   b) Print out to the terminal the contents of a file called `sometext.txt`. The file exists in the current directory.

   c) Print out to the terminal the last 5 lines of a plain text file called `sometext.txt`. The file exists in the current directory.

   d) Print out to the terminal the last 5 lines of *each* file that ends in the extension `.txt` and lives in the current directory

   e) Write a `for` loop which prints each integer from 0 to 6 (including 0 and 6).

3. Using *Euler* and the `module` command, answer the following questions.

    a) Are there any modules loaded (`module list`) when you log in on *Euler*?

    b) What version (version number) of `gcc` is available to you without loading any modules?

    c) List all `cuda` modules available on Euler.

    d) List one other piece of software that has a module on *Euler* and write one sentence about what it does. (If you aren't familiar with any of the other software, google one up and write a sentence about it.)

4. Write a bash script called `task4.sh` with a Slurm header which asks for

    - 2 CPU cores
    - A job name of `FirstSlurm`
    - An output file called `FirstSlurm.out`
    - An error file called `FirstSlurm.err`

    and runs a single command to print the hostname of the machine (compute node) running the job. This job should be submittable by running `sbatch task4.sh` on the head node.

5. Research some useful Slurm tools (one sentence responses):

    a) In what directory does a Slurm job on *Euler* begin execution? You may run some jobs in different directories to check this.

    b) Explain what `SLURM_JOB_ID` is in the environment of a running Slurm job.

    c) How would you track the status of job(s) run by yourself? Assume that the job(s) have not been completed yet.

    d) How would you cancel a job submitted by yourself? Assume that the job is still in the queue.

    e) Explain what the following script header line specifies: `#SBATCH --gres=gpu:1`

    f) (Optional) Explain what the following script header line specifies: `#SBATCH --array=0-9`

6. Write a C++ program called `task6.cpp` that:

   a) Takes a command line argument `N`.

   b) Prints out each integer from `0` to `N` (including `0` and `N`) in ascending order with the `printf` function.

   c) Prints out each integer from `N` to `0` (including `N` and `0`) in descending order with `std::cout`.

   For each printing process, the integers should be separated by spaces on a single line ending in a newline.

   - Compile command: `g++ task6.cpp -Wall -O3 -std=c++17 -o task6`
   - Run command: `./task6 N`
   - Expected output (followed by a newline):
     ```
     0 1 2 3 ··· N
     N ··· 3 2 1 0
     ```
   - Example expected output for `N` $= 6$ (followed by a newline):
     ```
     0 1 2 3 4 5 6
     6 5 4 3 2 1 0
     ```