High performance scientific computing in C++

June 20, 2021

Online couse infrastructure

- External participants: please download material as mentioned in the mail and prepare your set up.
- Regular participants, please login to the Jupyter-JSC system
- After logging in, try to add a new jupyterlab. Choose JUSUF as the system and training2119 as the project.
- For the partition choose **LoginNode**, and then start. Wait until the swirly things stop and you see the panel.
- What we will most need from there is the terminal, which should be at the bottom.
- In the terminal type this:

```
$ source $PROJECT/set_vars.sh
```

- After this, your paths should be set correctly. Test it using

```
$ g++ --version
$ clang++ --version
```

  You should see GCC version 11.1 and Clang version 12.0.

- The setup script must be run at the beginning of every new login to JUSUF for this course.
- It creates user specific working directories, downloads and updates course material and sets up the environment variables for compilers and libraries.
- After the script `setup.sh` is sourced, the following environment variables (EV) and additional shortcuts (SC) are available
  - `cxx2021` : (EV) Location of your private working area for the course
  - `swhome` : (EV) Top level folder for software installations for compilers and libraries
  - `cdp` : (SC) Change directory to the top level of your private workspace
  - `pathadd` : (SC) Prepend a new folder to PATH. E.g., `pathadd /x/y/z/bin`

  - `pathrm`: (SC) Remove a folder from PATH
  - `libpathadd`, `libpathrm`: (SC) Same as above, but for `LD_LIBRARY_PATH`, `LD_RUN_PATH`, `LIBRARY_PATH`
  - `incpathadd`, `incpathrm`: (SC) Same, but for `CPATH`, which is searched by the compilers for include files.
  - `cmpathadd`, `cmpathrm`: Same, but for `CMAKE_PREFIX_PATH`
  - `G`: (SC) Compiler wrapper for `g++` using common options `-std=c++20 -pedantic -Wall -O3 -march=native`
  - `A`: (SC) Similar to `G`, but for Clang. It also uses Clang's own implementation of the standard library, `libc++`
  - `B`: (SC) Similar to `A`. But it uses GCC's implementation of the standard library.

- The scripts `G`, `A` and `B` default to producing executables whose names are deduced from the names of the source files. E.g.: `G hello.cc` produces the executable `hello.g`, `A hello.cc` produces `hello.l` and `B hello.cc` produces `hello.b`
- They recognize libraries we need during the course: `G -tbb xyz.cc` compiles `xyz.cc` with suitable include and library options to use TBB
- The folder `yourworkspace/software`: Any software you build and install with this installation prefix will be found by the compilers
- Run simple compilation and small programs on the Login node, as you would on your laptop.
- For heavier workloads, we will use the batch system during the course. Run the executable `a.out` using 64 maximum threads on a JUSUF compute node as follows:
  `srun --nodes=1 --cpus-per-task=64 a.out [options]`
- For external participants: the path manipulation utilities used in the course are available with the course material in the file `code/bash/pathutils.sh`. It contains only BASH functions like `pathadd`, and nothing specific to our setup on JUSUF. Similarly, the scripts `G`, `A` and `B` can be found in `bin`.