

An Introduction to High Performance Computing: Exercises

Paul Sumption
support@hpc.cam.ac.uk

Research Computing Services (<http://www.hpc.cam.ac.uk/>)
University Information Services (<http://www.uis.cam.ac.uk/>)

22nd May 2018 / CSD3 User Training

Exercise 2: Transfer some files

You will need to transfer the exercise files to the cluster.

- ▶ Open a new Linux terminal on your training computer.
- ▶ Enter this command: `cd ~\Course_material`
- ▶ Check the file 'exercises.tar' is in your directory listing
- ▶ Hint: `ls`

Exercise 1: Login

Using a Linux terminal you will login to the cluster with your HPC training account.

- ▶ Start the terminal by double clicking on the terminal icon
- ▶ In your terminal enter:
 - ▶ `ssh -Y abc123@login-cpu.hpc.cam.ac.uk`
Replace abc123 with your training account username
 - ▶ Enter your password as supplied on the sheet
 - ▶ Leave this terminal open, you will need it for exercise 3,4 and 5!

Exercise 2: Transfer some files

Transfer the exercises.tar to your HPC home folder.

- ▶ In the local terminal on your training computer enter the command:
 - ▶ `sftp abc123@login-cpu.hpc.cam.ac.uk`
Change abc123 to your training account username
 - ▶ The command: `put exercises.tar` will transfer the file from your local computer to the remote one
 - ▶ Check the file 'exercises.tar' is in your directory listing
 - ▶ Hint: `ls`
 - ▶ Type 'exit' to close the local terminal

Exercise 3: Unzip the excercises.tar file

- ▶ In a terminal logged into the cluster: (see exercise 1.)
- (a) Use the `ls` to list your home folder contents — you should see the copy of `exercises.tar`.
Hints: Do `cd ~` then `ls -al`. Note that `cd ~` will take you back to your home directory.
- (b) Unpack the tar archive to create an exercise subdirectory.
Hints: Do `tar -xvf exercises.tar`
- (c) Move the exercise subdirectory to a new directory.
Hints: Do `mv exercises myexercises`

Exercise 4: File listings

- ▶ In a terminal logged into the cluster: (see exercise 1.)
- ▶ [(a)] List the contents of your current directory `ls`. This won't show everything — use `ls -al` for a long listing showing all files. Initially you will start in your home directory — use `pwd` to print the name of your current working directory. If you get lost, you can always do `cd` without arguments to return to your home directory.
- (b) Focus your long listing on all files with names beginning “myexercises”.
Hints: Do `ls -al myexercises*`
- (c) Print a long listing of the subdirectory `myexercises`.
Hints: Do `ls -al myexercises/`.

Exercise 5: Learn more about a command

- ▶ In a terminal logged into the cluster: (see exercise 1.)
- (a) View the man page for the `cp` command by doing `man cp`. Use `SPACE` to page down and `b` to page up. Press `q` to exit the manual page command.
- (b) View the man pages for the `mkdir` and `mv` commands.

Exercise 6: Environment Modules

- ▶ Connect to the cluster using your training account: See exercise 1 if you have closed your terminal.
- ▶ Get a list of modules that are currently loaded
Hints: `module list`
- ▶ Get a list of available R modules
Hints: `module av r`

Exercise 7: Run an Rscript

- ▶ Connect to the cluster using your training account: See exercise 1 if you have closed your terminal.
- ▶ In the exercises folder you transferred earlier there is a file called `test.r`
- ▶ Run this script using: `Rscript hello.r`
- ▶ Load the module for: `r-3.4.3-gcc-5.4.0-rbvhnaga`

Hints: `module load r-3.4.3-gcc-5.4.0-rbvhnaga`

- ▶ Run the script again: `Rscript hello.r`
- ▶ What happens? what changes?

Exercise 8: Explained

- ▶ To load one of our R modules: `module load r/(version)`
- ▶ `echo " "` outputs the text between the quotes, `>` redirects the text into the `.Renviron` file.
- ▶ When we start R the `.Renviron` file is read and R will now be aware of our local library directory.
- ▶ `.libPaths()` is how to check your library locations

Exercise 8: Install the R library locally

As a user you can create a local R library directory for packages that you want to install.

- ▶ Load an R module: `module load r-3.4.3-gcc-5.4.0-rbvhnaga`
- ▶ Create a folder in your home for your own R package installs:
`mkdir ~/my-R-libs`
- ▶ Make R aware of the new library location:
`echo "R_LIBS_USER=~ /my-R-libs" > ~/.Renviron`
- ▶ Start R: `R`
- ▶ Display your library paths: `.libPaths()`
- ▶ Try loading a library: `require(pander)`
- ▶ Its not insalled, lets install it: `install.packages("pander")`
- ▶ Try loading a library: `require(pander)`
- ▶ Library is now installed, lets quit R: `quit()`

Exercise 9: Modules and Compilers

- ▶ Connect to the cluster using your training account: See exercise 1 if you have closed your terminal.
- ▶ Go to the `myexercises` directory.

Try to compile the `hello.c` program using the default `gcc` compiler (it will fail because there is a deliberate bug).

Hints: `gcc hello.c -o hello`

- ▶ To fix the problem, open the `hello.c` file in the `gedit` editor.

Hints: Launch `gedit` in the background by doing `gedit&`. A `gedit` window should appear. Remove the word `BUG`, save the file and recompile. Do `./hello` to run the program.

- ▶ If you get this error:
`WARNING **: cannot open display:`
then you have missed the `'-Y'` in your SSH command

Exercise 10: Modules and Compilers

- ▶ The default version of gcc is 4.8.5. Compile hello.c again with `gcc 5.4.0`.

Hints: `module av`, `module load gcc-5.4.0-gcc-4.8.5-fis24gg`, then `gcc hello.c -o hello2`

README: Job Scripts!

If you are attending the HPC Course:

- ▶ Use `job_script` for the exercises when running the exercises during the course
- ▶ Use `slurm_submit.peta4-skylake` if running the exercises outside of the course
- ▶ I recommend you make a copy of the job script for each exercise

Exercise 11: Submitting a Matlab job

- ▶ Submit a job which will run `matlab` on the `file.m` command file (which contains just the `ver` command).

Hints:

1. Edit the `job_script` in your exercises directory.
2. In the module load section: `module load matlab/r2017b`
3. Set the value of application to `"matlab -nodesktop -nosplash -nojvm"`
4. Set the value of options to `"-r file"`
5. Submit the job with `sbatch job_script`. The jobid is then printed.
6. Watch the job in the queue with `squeue`.
7. After it has disappeared, open the output file `slurm-jobid.out` in your editor. It should contain a list of licensed Matlab features.
8. For more demanding work you can increase the available memory by increasing the number of cpus.

Exercise 12: Submitting compiled code

- ▶ Submit a job which will run a copy of your hello program on 1 cpu.

Hints:

1. Edit the script `job_script` in your exercises directory.
Set:
`#SBATCH --nodes=1`
`#SBATCH --ntasks=1`
`application="./hello"`
2. Submit the job with `sbatch job_script`. The jobid is then printed.
3. Watch the job in the queue with `squeue`.
4. After it has disappeared, open the output file `slurm-jobid.out` in your editor. There should be exactly one "Hello, World!" message.

Exercise 13: Array Jobs

- ▶ Submit your last job in the form of an array with indices 1-32. Use `-H` with `sbatch` to mark the array as held (so that it won't run immediately).

Hints:

1. Use `sbatch -H --array=1-32 job_script`
2. Use `squeue -u userid` to see your array job. Note that `-r` reports each array element individually.

- ▶ Release array element 1 and allow it to run. Then release the others.

Hints:

1. Use `scontrol release ${SLURM_ARRAY_JOB_ID}_1`
2. Use `squeue -u userid` again to watch what happens.
3. Release the others with
`scontrol release ${SLURM_ARRAY_JOB_ID}`
i.e. use the array id to release the entire array.
4. When all the jobs complete you should have 32 `slurm-${SLURM_ARRAY_JOB_ID}_N.out` files saying hello from various cpus on possibly multiple nodes.