ECMWF training course – 2017
Compiling environment – ecgate

Practical examples

N O T E S:

1. Use "gfortran"
2. **Use ecgate - not** your local desktop system.
3. Work interactively or in batch mode (using sbatch and the '.cmd' files, when available).
4. See slides, man pages or online documentation.
5. Some job examples are available under: https://software.ecmwf.int/wiki/display/UDOC/Slurm+job+script+examples
6. Create a subdirectory for this practical session, e.g.

```
% cd $SCRATCH
% tar xf ~trx/intro/ecgate-compiling-practicals.tar.gz
% cd ecgate-compiling-practicals
```

7. A single Makefile is available for all the exercises below.

EXERCISE 1 (Compiling, linking)

Can you try to write, compile and run the famous "Hello whoever" program? A sample program 'hello.f' is available in the directory.

Try to compile the code manually with gfortran. After that, run 'make hello' to compile the code. Did you use the same line as in the Makefile?

EXERCISE 2 (Creation of libraries)

Can you adapt primecheck.cmd to create (and use) your own library containing the function "prime"? The files primecheck.f and prime.f are available in the directory. Edit the makefile to include the code to produce the static library and the executable. The compilation and linking should then work when running 'make primecheck'.

ADVANCED: When that works, try to produce and use the dynamic library instead. Does it run? If not, how can you fix it?

EXERCISE 3 (Usage of libraries)

The ecCodes library is used to decode GRIB data, e.g., extracted from MARS. This library is called in Retrieve_decode_grib_api.cmd. BUFR data will be decoded using the same library. See job example Retrieve_decode_bufr.cmd. The sample data files are available (grib_file contains the GRIB data and bufr_file the BUFR data).

The source code files grdemo.f90 and bfdemo.f90 are available in the directory. If you use the Makefile, run ‘make grdemo’ or ‘make bfdemo’.

Can you compile (and run) the GRIB and/or BUFR decoding programs, modifying the Makefile with the appropriate instructions?

Note that BUFR decoding will be covered later this week.

EXERCISE 4 (Basic Debugging)

Please compile and run the program ‘debug.f’. Again, try to include the required code to be able to compile the code with ‘make debug’. How would try to stop this program earlier or get more information about the problem? Can you get a core file? How would you inspect it?

EXERCISE 5 (Profiling – ‘optimisation’)

Getting back to the second exercise (codes prime.f and primecheck.f), can you measure how long the program takes to run?

Which compilation option could you try out first to speed up this program? Can you try to use it?

You may want to try to generate a profile of the application and read with gprof.

EXTRA – for those who finish early (Profiling – ‘optimisation’)

Get the NAS serial benchmark tar here: **/home/ectrain/trx/intro/NPB3.3-SER.tar.gz**

Now, uncompress it and try to figure out how to build at least one of the tests with the class “A” size.

Then, experiment with different optimisation flags and compare the timings. Try and generate a profile of the application to read with gprof. What are the routines taking up most of the time? What are the ones called the most?