ABINIT School 2019
January 21 – 25, 2019
Bruyères-le-Châtel, France

Installing ABINIT on a Supercomputer – Hands-on session

By following this step-by-step procedure, you should understand how to install ABINIT on a supercomputer using the **module** command.
In the following, all the commands are those of the CEA **Cobalt** supercomputer (*TGCC* French computing center, 2019).

Open a terminal.
First of all, clean your environment (unload all "modules") by typing:

```
module purge
```

It is mandatory to have a Fortran compiler and a MPI library. Load the corresponding modules (Intel compiler suite and MPI):

```
module load intel mpi
```

A linear algebra library (`blas/lapack`) is also needed for ABINIT. On a supercomputer, it is mandatory to have *optimized* library. It is also mandatory to use a *multithreaded* library (taking advantage of a multi-task parallelism). It is also better to have a *parallel* library (based on `Scalapack`).
On the `Cobalt` supercomputer, the following modules should be loaded:

```
module load feature/mkl/multi-threaded
module load scalapack
```

In the directory dedicated to this tutorial, you should find the abinit-x.y.z.tar.gz file. If not, download this file from ABINIT website (www.abinit.org) and then extract it.
Enter in the abinit-x.y.z directory.

It is highly recommended to create a separate directory to compile the code and store the object files. Let's create the build directory by typing:

```
mkdir build
```

Enter in the build directory.

Let's first try to configure the build automatically.
Just type (in the abinit-x.y.z/build folder):

```
../configure
```

After a few seconds, you should obtain the configuration report.

What do we see in it (usual case)?

```
* OpenMP enabled : no
* MPI    enabled : no
* TRIO   flavor = none
* DFT    flavor = none
```

The parallel build (MPI and *openMP*) is not activated; no plugin has been activated (no "Transferable I/O" (TRIO) plugin, no DFT plugin).
This is not optimal! Let's change this.

First of all, we will create a *configuration file*; this is more convenient than having a long command line to configure the code.

Type the command (in the terminal):

```
hostname
```

You should get the name of the computer.
Something like: computername or computername.domain.domain
Let's create a file named computername.ac in the build directory. Use your favorite editor for that.

Then, in this configuration file, add the following lines, and retry ../configure:

```
enable_mpi="yes"
enable_openmp="yes"
```

If the configuration script goes to the end, MPI has been detected.
If not, the best practice, on a supercomputer, is to use a *parallel compiler*. For that, add the following lines in the configuration file:

```
FC="mpif90"
CC="mpicc"
CXX="mpicxx"
```

If not, you should add a line locating the MPI installation.

> Note: another possibility is is to locate the mpif90 file.
> Type: `which mpif90`. You should find it in a place like:
> `path_to_MPI/bin/mpif90`
> Then add the following line in the configure to use directly mpif90 as Fortran compiler. For that, add the following lines in the configuration file: `with_mpi_prefix="path_to_MPI"`

This time, everything should be OK; MPI has been detected.

Unfortunately, the linear algebra library detection is not optimal. The configure script must be helped a little. Add the following lines it. The first one tells the script to select the *Intel Math Kernel Library* (MKL) we have

previously loaded. The second one uses a specific variable to locate the library (specific to `Cobalt` computer):

```
with_linalg_flavor="mkl+scalapack"
with_linalg_libs=${SCALAPACK_LDFLAGS}
```

Now, let's try to add the ABINIT plugins.

For this tutorial, let's first try to add **`netCDF`** ("Transferable I/O"=TRIO plugin). Add the following in the configuration file and re-run the configure script:

```
with_trio_flavor="netcdf"
```

If the configuration script goes to the end, you should see the following in the configuration report:

```
* TRIO flavor = netcdf
```

If not, you just have to load the `netcdf` module:

```
module load netcdf-fortran
```

and help the `configure` script to locate the library (yes!, these lines are not so simple to guess). Add the following in the configuration file:

```
with_netcdf_libs="-L${NETCDFC_ROOT}/lib -lnetcdf \
                  -L${NETCDFFORTRAN_ROOT}/lib -lnetcdff"
with_netcdf_incs="-I${NETCDFC_ROOT}/include \
                  -I${NETCDFFORTRAN_ROOT}/include"
```

At this stage, the detection of *netCDF* should be OK.

Let's now add one additional plugin: *libXC* (library of exchange-correlation functionals).
First load the `libXC` module:

```
module load libxc
```

Then help the `configure` script to locate the library (add the following lines in the configuration file):

```
with_dft_flavor="libxc"
with_libxc_libs="-L${LIBXC_ROOT}/lib -lxc -lxcf90"
with_libxc_incs="-I${LIBXC_ROOT}/include"
```

In the final configuration report, you should see this:

```
* DFT flavor = libxc
```

Everything is almost ready for the compilation
Just type:

```
make mj4
```

...and wait... wait..

At the end of the compilation process, you should get some messages explaining how to check the installation by running the automatic tests.

Let's try if ABINIT can run; just type:

```
cd tests && ../../tests/runtests.py fast
```

And wait for the report.
Many other automatic tests are available. You can get the list with:

```
../../tests/runtests.py -show-info
```

ABINIT executable is located in a specific directory. You can it here:

```
abinit-x.y.z/build/src/98_main/abinit
```

OK, this is done.
You have compiled ABINIT, congratulation!

To go further...

Improving FFT implementation

On the same model as Linear Algebra, you can take advantage of an optimized `Fast Fourier Transform` (FFT) library.
The FFTW library (included in the *Intel Math Kernel Library*) can be loaded as follows:

```
module load fftw3/mkl
```

Then add the following lines in the configuration file (to help ABINIT to locate the library):

```
with_fft_flavor="fftw3"
with_fft_incs="-I${MKL_INCDIR}"
with_fft_libs=${MKL_LDFLAGS}with_netcdf_libs="-
```

A possible configuration file (**Cobalt** supercomputer)

```
FC="mpif90"
CC="mpicc"
CXX="mpicxx"

enable_mpi="yes"
enable_openmp="yes"

with_linalg_flavor="mkl+scalapack"
with_linalg_libs=${SCALAPACK_LDFLAGS}

with_fft_flavor="fftw3"
with_fft_incs="-I${MKL_INCDIR}"
with_fft_libs=${MKL_LDFLAGS}

with_trio_flavor="netcdf"
with_dft_flavor="libxc"

with_libxc_libs="-L${LIBXC_ROOT}/lib -lxc -lxcf90"
with_libxc_incs="-I${LIBXC_ROOT}/include"

with_netcdf_libs="-L${NETCDFC_ROOT}/lib -lnetcdf \
                  -L${NETCDFFORTRAN_ROOT}/lib -lnetcdff"
with_netcdf_incs="-I${NETCDFC_ROOT}/include \
                  -I${NETCDFFORTRAN_ROOT}/include"
```

To compile, the following modules have to be loaded:

```
module load feature/mkl/multi-threaded
module load intel mpi
module load scalapack fftw3/mkl
module load netcdf-fortran libxc
```