FROM RESEARCH TO INDUSTRY

ABINIT Hands-on 2019

A newcomer-oriented school to ab initio nanoscience simulations

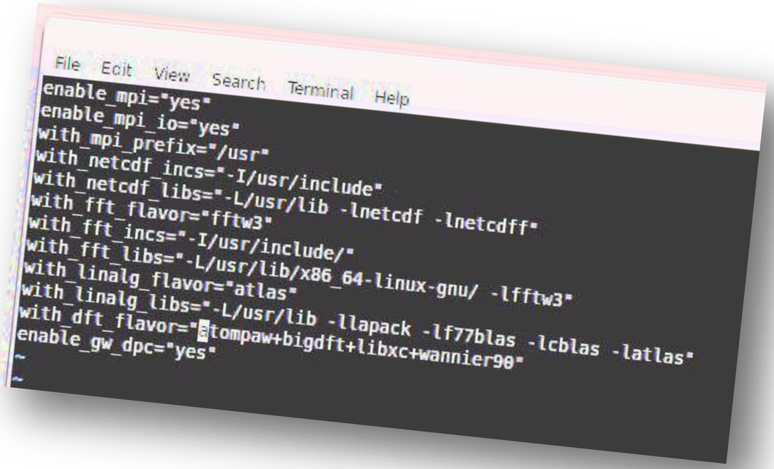
January 21-25, 2019 - Bruyères-le-Châtel, France

INSTALLING ABINIT

LAPTOPS – WORKSTATIONS - SUPERCOMPUTERS

Marc Torrent

CEA, DAM, DIF, France



```
enable_mpi="yes"
enable_mpi_io="yes"
with_mpi_prefix="/usr"
with_netcdf_incs="-I/usr/include"
with_netcdf_libs="-L/usr/lib -lnetcdf -lnetcdff"
with_fft_flavor="fftw3"
with_fft_incs="-I/usr/include/"
with_fft_libs="-L/usr/lib/x86_64-linux-gnu/ -lfftw3"
with_linalg_flavor="atlas"
with_linalg_libs="-L/usr/lib -llapack -lf77blas -lcblas -latlas"
with_dft_flavor="atompaw+bigdft+libxc+wannier90"
enable_gw_dpc="yes"
~
~
```

www.cea.fr

OUTLINE

ABINIT INSTALLATION BASICS

WHAT DO YOU NEED?... AT LEAST

- A ***Linux*-like environment** (*Linux* distribution, *macOS*, …)
  Windows accessible via
  1- Linux-under-windows (*cygwin*, *minGW*, …)
  2- An integrated environment (*visual****)

- A **compiler suite**, at least Fortran-2003 and C
  Some features only available if Fortran 2008, C++, Cuda

- A **MPI** library (Message Passing Interface)
  Not mandatory but strongly recommended

- A “**BLAS/LAPACK**” library (linear algebra)
  Can be downloaded *on the fly* if Internet connection

- ABINIT ***tarball*** file
  Downloadable from www.abinit.org

- **Internet** connection ?
  Can be convenient to download “on the fly” some extra packages

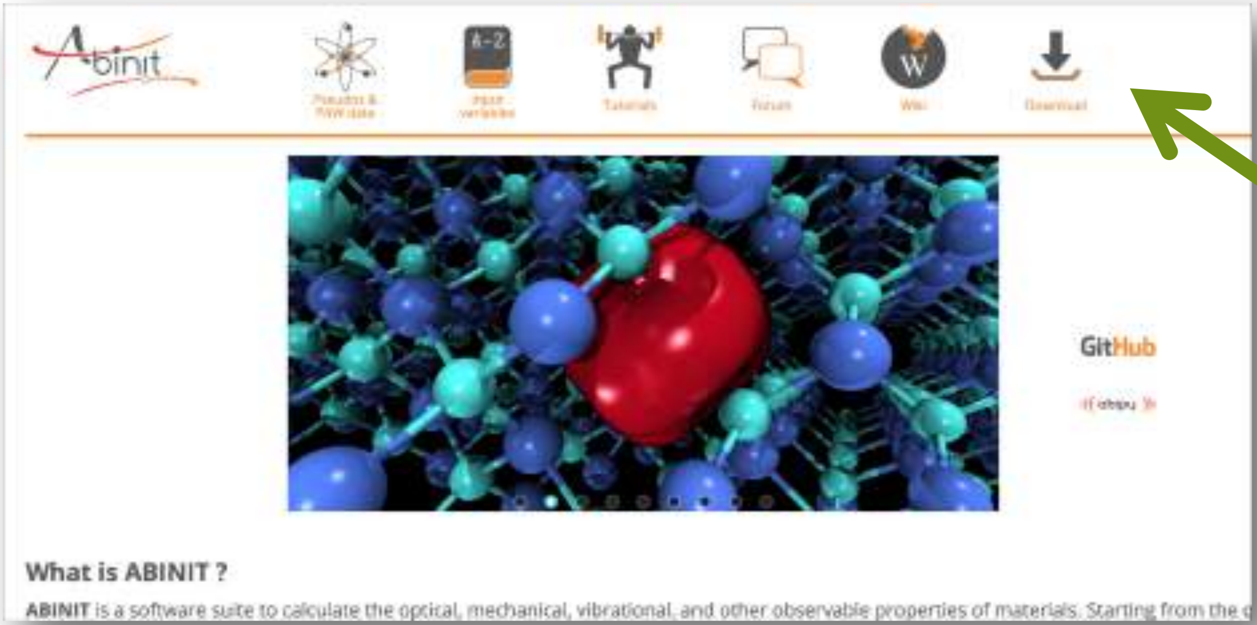
THE COMPILATION PROCEDURE

- Getting the source files
- Getting additional plugins (or fallbacks)
- Configuring the software
  Making the build ready on your specific system
  - *Most of the process is automatic*
  - *Additional plugins can be specified*
  - *Location of external libraries has to be specified*
- Building
- Installing
  Copying to the final destination

Critical step !

GETTING THE INSTALLATION FILE (TARBALL)

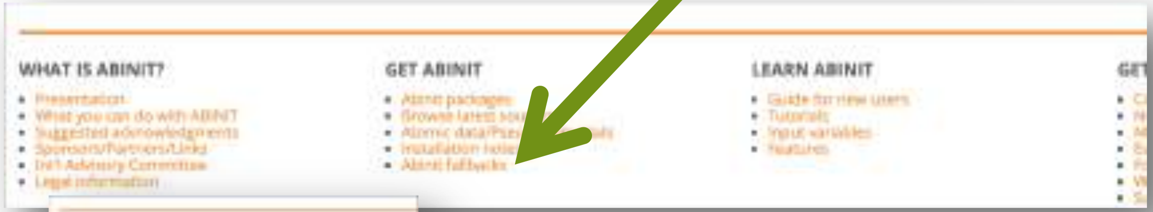
- The source package



What is ABINIT ?

ABINIT is a software suite to calculate the optical, mechanical, vibrational, and other observable properties of materials. Starting from the

- The additional plugins “Fallbacks” are provided on ABINIT web site

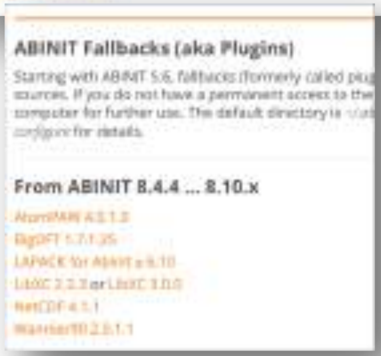


WHAT IS ABINIT?
- Presentation
- What you can do with ABINIT
- Suggested acknowledgments
- Sponsors/Partners/Links
- Int'l Advisory Committee
- Legal information

GET ABINIT
- Abinit packages
- Browse latest sou
- Atomic data/Pse
- Installation note
- Abinit fallbacks

LEARN ABINIT
- Guide for new users
- Tutorials
- Input variables
- Features

GET



ABINIT Fallbacks (aka Plugins)

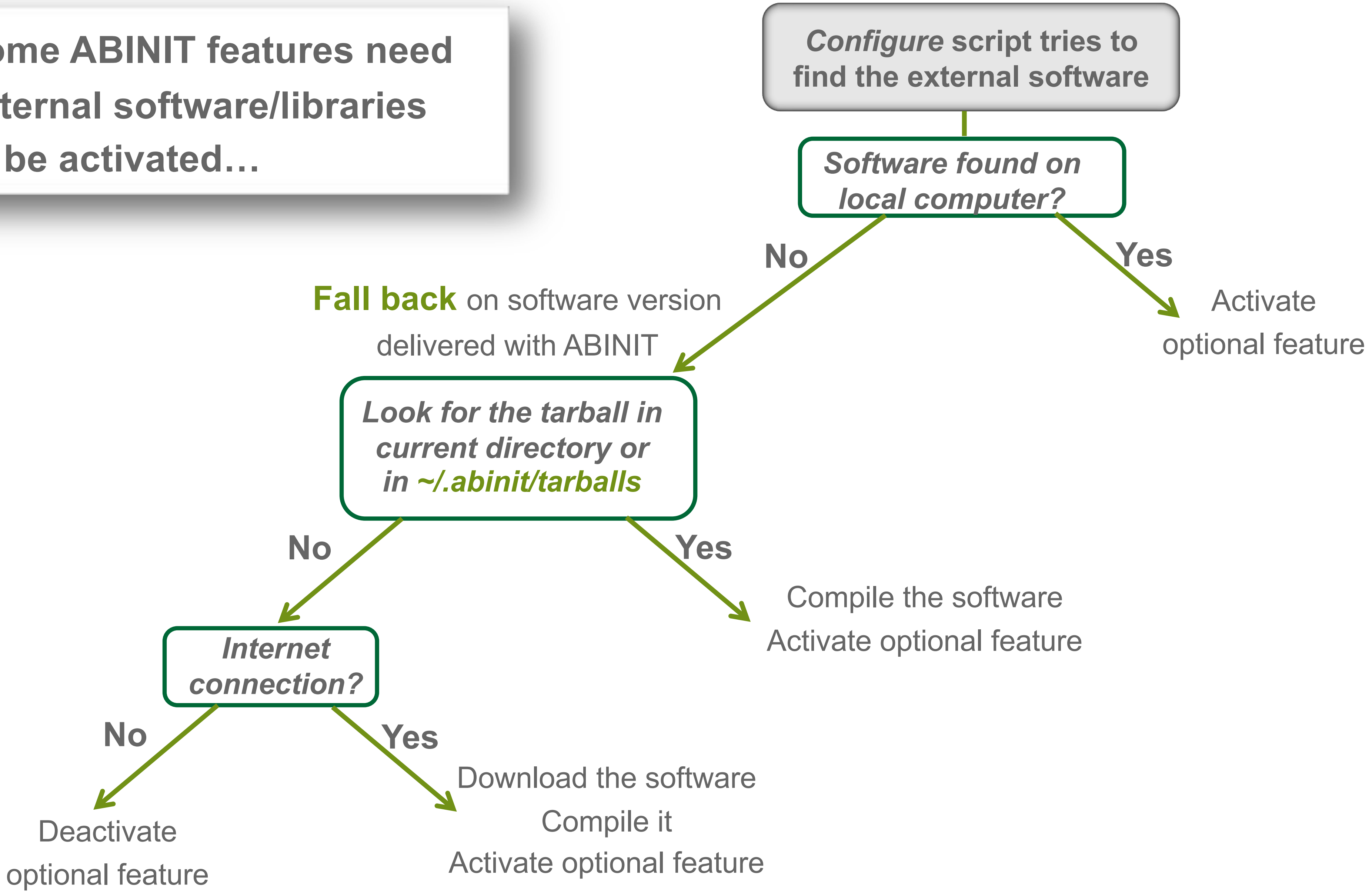
Starting with ABINIT 5.6, fallbacks (formerly called plug
sources. If you do not have a permanent access to the
computer for further use. The default directory is
configure for details.

From ABINIT 8.4.4 ... 8.10.x

AtomPAW 4.1.1.0
BigDFT 1.7.1.25
LAPACK for ABINIT ≥ 8.10
LibXC 2.2.3 or LibXC 3.0.0
NetCDF 4.1.1
Wannier90 2.0.1.1

WHAT ARE THE PLUGINS/FALLBACKS?

Some ABINIT features need external software/libraries to be activated…



Configure script tries to find the external software
Software found on local computer?
No
Yes
Activate optional feature
Fall back on software version delivered with ABINIT
Look for the tarball in current directory or in ~/.abinit/tarballs
No
Yes
Compile the software
Activate optional feature
Internet connection?
No
Yes
Deactivate optional feature
Download the software
Compile it
Activate optional feature

WHAT ARE THE FALLBACKS?

- Mandatory
  - **Blas/LAPACK** : Linear Algebra
    A vendor library strongly recommended
    Fallback version not efficient
- Almost mandatory
  - **netCDF/netCDF-Fortran** : to write machine-independent binaries
    Used by post-processing tools, trajectory restart, …
  - **LibXC** : a collection of Exchange-Correlation functionals
    If not activated, on a few XC functionals available
- Optional
  - **Wannier90**: use of Maximally Localized Wannier Functions
    Used by post-processing tools (transport properties)
  - **bigDFT** : to activate the possibility to use a wavelet basis
  - **AtomPAW** : the PAW atomic data generator

CONFIGURING

```
../configure --help
```

- Most of the properties of the environment are **automatically detected**
- If the environment is not compatible with a given feature, the latter is automatically deactivated
- Presence of ***Plugins*** is automatically checked
  It is possible to enforce the use of a fallback (and compile it on the fly)
- It is possible to specify the **destination** for the executables
- It is possible to activate some specific **architecture-dependent flags**: use of shared memory (*openMP*), use of a Graphical card, …)

INSTALLING

`make install`

- Several executable files are copied into the destination directory :
  - **abinit** : *main executable.*
    All-in-one software : DFT, DFPT, DMFT, MBPT, PIMD, NEB, …
  - **cut3d** : *post-processing tool : extracting data,*
    converting ABINIT output files into common data format
  - **anaddb** : *mandatory in the case of response function calculation ;*
    ANAlysis of the Derivative DataBase
  - **conducti** : *transport properties (conductivity, reflectivity, linear optics)*
  - **aim** : *Bader Atom-in-Molecule analysis*
  - **macroav** : *macroscopic average technique applied on potentials*
  - **multibinit** : *second-principles approach for lattice dynamics*
  - **tdep** : *response function and thermodynamics including temperature*

SPECIFIC ARCHITECTURE: LINUX DISTRIBUTION
UBUNTU, REDHAT, …

- Most of the compilers and libraries are available by default or as packages
  ```
  sudo apt-get install gfortran openmpi
  ```
- netCDF is available as a package
  ```
  sudo apt-get install netcdf
  yum –y install netcdf
  ```
- Some debian packages or RPM are available on the libXC’s homepage
- No recent ABINIT version available directly as a Linux package
- A single “configure” is usually OK to compile directly.

SPECIFIC ARCHITECTURE : MACOS

Automatic method: using macports package manager

- Install macport
  See http://www.macports.org
- Install abinit
  `sudo port install abinit`

Automatic method: using homebrew package manager

- Install `homebrew`
  Everything explained here: http://brew.sh
- Install abinit
  `brew install brewsci/science/abinit`

Manual method: compile by yourself

- Need to install a Fortran compiler and MPI library
- Need to compile netcdf and libxc first
- See the rest of this presentation

COMPUTING CENTERS (SUPERCOMPUTERS)

- ABINIT is installed in most computing centers.
  If not, ask the system administrator

- The “module” command is now widely used to load environments and software.

  ```
  module load abinit
  ```

- Compiling ABINIT on supercomputers or small computer clusters is made easy by the “module” command.
  But configuring the build is tricky
  See later in the presentation

- Internet connection is usually not available.
  Optional fallbacks have to be preloaded.

HOW TO OBTAIN AN ABINIT EXECUTABLE

CONFIGURING AND COMPILING – GOOD PRACTICES

- Extract the archive and enter the directory

```
tar –xvzf abinit-x.y.z.tar.gz
cd abinit-x.y.z
```

- Create a working directory and enter it

```
mkdir build
cd build
```

- Configure, according to your needs and computer

```
../configure [options]
```

- Compile

Always use parallel build with predefined make commands

```
    make mj4        >> 4 tasks in parallel
or  make mj8        >> 8 tasks in parallel
```

- Create a working directory and enter it

```
make install
```

This step is the most important

CONFIGURING AND COMPILING – GOOD PRACTICES

- Applying the previous procedure, you always get ABINIT executable files
- But:
  - *They are installed in /usr/local*
  - *Parallel features (MPI, openMP or GPU) are not necessarily used*
  - *Optional features (plugins/fallbacks) are not necessarily activated*
  - *Elementary functions (i.e. linear algebra, FFT) can be inefficient*

- ABINIT can be used on a laptop but <u>you can do better</u>
- It is completely inadequate for a parallel computer

- The configure step has to be tuned

ABINIT CONFIGURATION – HOW TO CUSTOMIZE?

- Choose the destination for the executable file

```
../configure --prefix=destination_directory
```

- Choose the Fortran compiler

```
../configure FC=mpif90
```

ABINIT CONFIGURATION – HOW TO IMPROVE?

- First step : look at the messages at the end of the configuration

```
===========================================================
==== Final remarks                                      ===
===========================================================

Summary of important options:

  * C compiler      : gnu version 6.2
  * Fortran compiler: gnu version 6.2
  * architecture    : unknown unknown (64 bits)

  * debugging       : basic
  * optimizations   : standard

  * OpenMP enabled  : no (collapse: ignored)
  * MPI    enabled  : yes
  * MPI-IO enabled  : yes
  * GPU    enabled  : no (flavor: none)

  * TRIO   flavor = netcdf
  * TIMER  flavor = abinit (libs: ignored)
  * LINALG flavor = netlib (libs: user-defined)
  * ALGO   flavor = none (libs: ignored)
  * FFT    flavor = none (libs: ignored)
  * MATH   flavor = none (libs: ignored)
  * DFT    flavor = libxc

Configuration complete.
```

ABINIT CONFIGURATION – HOW TO IMPROVE?

- First step : be sure to build a parallel executable

```
===========================================================
==== Final remarks                                      ===
===========================================================

Summary of important options:

  * C compiler      : gnu version 6.2
  * Fortran compiler: gnu version 6.2
  * architecture    : unknown unknown (64 bits)

  * debugging       : basic
  * optimizations   : standard

  * OpenMP enabled  : no (collapse: ignored)
  * MPI    enabled  : no
  * MPI-IO enabled  : no
  * GPU    enabled  : no (flavor: none)

  * TRIO   flavor = netcdf
  * TIMER  flavor = abinit (libs: ignored)
  * LINALG flavor = netlib (libs: user-defined)
  * ALGO   flavor = none (libs: ignored)
  * FFT    flavor = none (libs: ignored)
  * MATH   flavor = none (libs: ignored)
  * DFT    flavor = libxc

Configuration complete.
```

ABINIT CONFIGURATION – HOW TO IMPROVE?

- **First step : be sure to build a parallel executable**

```
configure –enable-mpi FC=mpif90
```

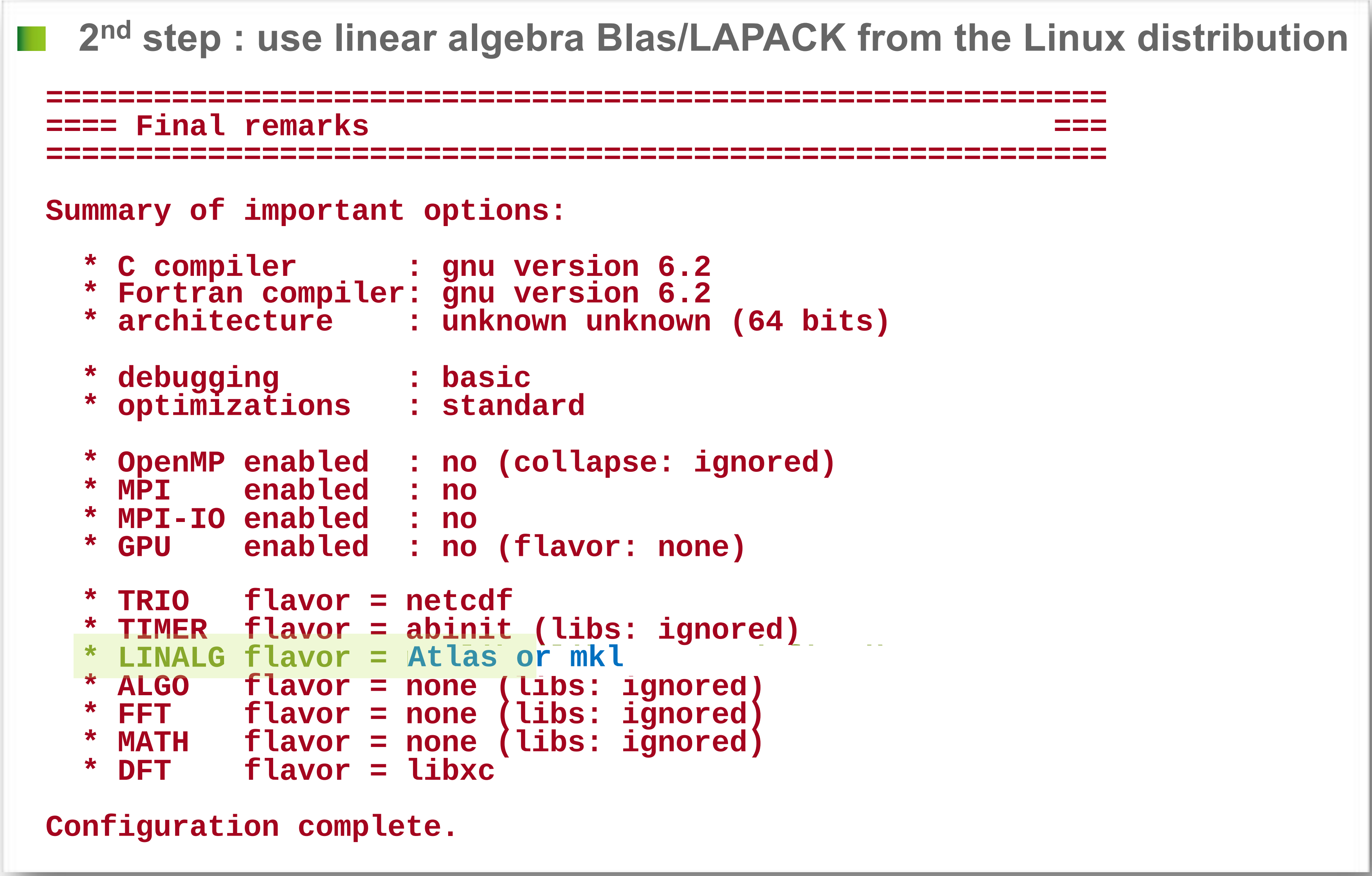
In some cases, the configure script does not find
the MPI library and/or executable ; how to help it…

```
configure --enable-mpi --with-mpi-prefix=path_to_mpi
```

where to find `bin/mpif90` and `include/mpif.h`

ABINIT CONFIGURATION – HOW TO IMPROVE?

- 2nd step : use linear algebra Blas/LAPACK from the Linux distribution



```
===========================================================
==== Final remarks                                     ===
===========================================================

Summary of important options:

  * C compiler       : gnu version 6.2
  * Fortran compiler: gnu version 6.2
  * architecture     : unknown unknown (64 bits)

  * debugging        : basic
  * optimizations    : standard

  * OpenMP enabled   : no (collapse: ignored)
  * MPI    enabled   : no
  * MPI-IO enabled   : no
  * GPU    enabled   : no (flavor: none)

  * TRIO   flavor = netcdf
  * TIMER  flavor = abinit (libs: ignored)
  * LINALG flavor = Atlas or mkl
  * ALGO   flavor = none (libs: ignored)
  * FFT    flavor = none (libs: ignored)
  * MATH   flavor = none (libs: ignored)
  * DFT    flavor = libxc

Configuration complete.
```

ABINIT CONFIGURATION – HOW TO IMPROVE?

- **2nd step : use linear algebra Blas/LAPACK from the Linux distribution**

If not automatically detected, enforce the use of it

On a laptop or a personal computer, default version is OK:

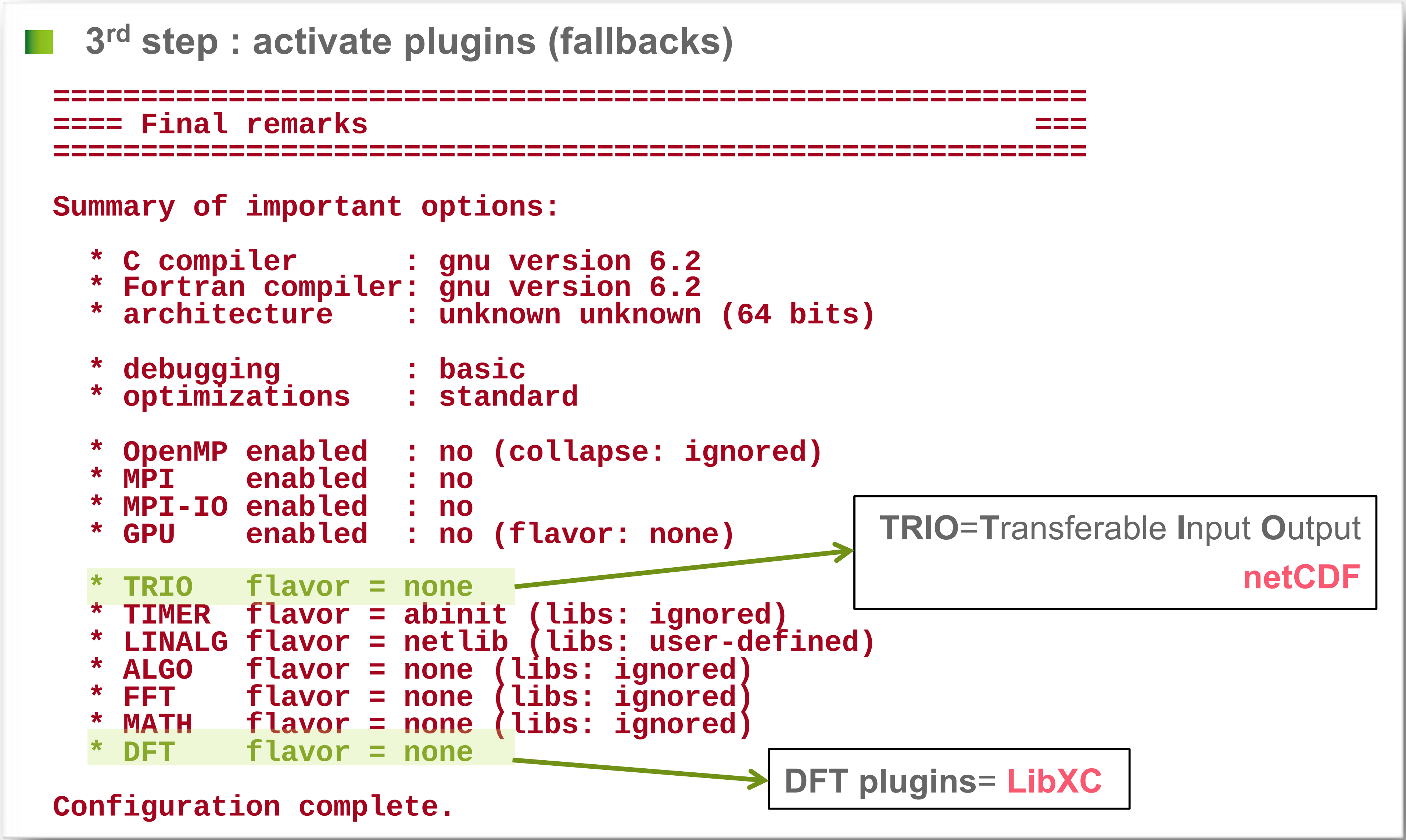
```
configure --with-linalg-libs=“-L/usr/lib –lblas –llapack”
```

ATLAS is a freely distributed library (available in most distributions):

```
configure --with-linalg-libs=“-L/usr/path_to/lib \
                -llapack -lf77blas -lcblas –latlas”
```

ABINIT CONFIGURATION – HOW TO IMPROVE?

- 3rd step : activate plugins (fallbacks)



```
==============================================================================
==== Final remarks                                                       ===
==============================================================================

Summary of important options:

  * C compiler       : gnu version 6.2
  * Fortran compiler: gnu version 6.2
  * architecture     : unknown unknown (64 bits)

  * debugging        : basic
  * optimizations    : standard

  * OpenMP enabled   : no (collapse: ignored)
  * MPI    enabled   : no
  * MPI-IO enabled   : no
  * GPU    enabled   : no (flavor: none)

  * TRIO   flavor = none
  * TIMER  flavor = abinit (libs: ignored)
  * LINALG flavor = netlib (libs: user-defined)
  * ALGO   flavor = none (libs: ignored)
  * FFT    flavor = none (libs: ignored)
  * MATH   flavor = none (libs: ignored)
  * DFT    flavor = none

Configuration complete.
```

TRIO=**T**ransferable **I**nput **O**utput
netCDF

DFT plugins= **LibXC**

ABINIT CONFIGURATION – HOW TO IMPROVE?

- 3rd step : activate plugins (fallbacks)

1-Ask for them !

```
configure   --trio-flavor="netcdf" \
            --dft-flavor=="libxc"
```

Everything found on the disk! →

```
* TRIO    flavor = netcdf
* DFT     flavor = libxc
```

OK

Some plugins not found but we "fall back" on packages accessible via Internet! →

```
* TRIO    flavor = netcdf
* DFT     flavor = libxc-fallback
```

OK

One plugin not found and no internet connection →

```
Error message!
```

ABINIT CONFIGURATION FILE

- Finally, the command line for the configuration step can be long:

```
configure   --with-trio-flavor="netcdf" \
            --with-dft-flavor="libxc" \
            --prefix=destination_directory \
            --enable-mpi FC=mpif90 \
            --with-mpi-incs="-Ipath_to_incs" \
            --with-mpi-libs="-Lpath_to_libs –lmpi.." \
            --with-linalg-libs="-L/usr/lib –lblas –llapack"
```

- It is possible to store all the options in **a configuration file.**

  The configuration script looks for it as

  1- name_of_computer.ac in $HOME/.abinit/build

  2- name_of_computer.ac in current directory

  3- any file given on command line :

  ```
  configure --with-config-file=name_of_file
  ```

COMMAND LINE VS CONFIGURATION FILE

```
configure  --with-trio-flavor=“netcdf”
           --with-dft-flavor=“libxc”
           --prefix=destination_directory
           --enable-mpi FC=mpif90
           --with-mpi-incs=“-Ipath_to_incs”
           --with-mpi-libs=“-Lpath_to_libs –lmpi..”
           --with-linalg-libs=“-L/usr/lib –lblas –llapack”
```

…is equivalent to…

```
configure --with-config_file=...
```

Configuration file

```
FC = mpif90
with_trio_flavor = “netcdf”
with_dft_flavor = “libxc”
prefix = destination_directory
enable_mpi = “yes”
with_mpi_incs = “-Ipath_to_incs”
with_mpi_libs = “-Lpath_to_libs –lmpi..”
with_linalg_libs = “-L/usr/lib –lblas –llapack”
```

Suppress “- -”
Replace “-” by “_”

HOW TO OBTAIN
AN EFFICIENT ABINIT EXECUTABLE

PARALLEL COMPUTERS

SUPERCOMPUTERS + ABINIT

- Favor the use of the **configuration file**
  Command line could be very long

- **Load the "modules"** (module load …)
  Before the compilation
  Before the execution

- Efficiency on supercomputers implies:
  - use of **hybrid parallelism** *(MPI+openMP)* ***IMPORTANT!***
  - use of preinstalled **vendor libraries** *(linear algebra, FFT)*
  - use of **parallel** and **multi-thread** versions of **libraries**

1- ACTIVATE OPENMP (MULTITHREAD)

MANDATORY !

- Add:

  `--enable-openmp` or `enable_openmp="yes"`

- Activate "multi-threaded" versions of libraries

  Examples:

  Intel mkl library: `"-lmkl_gnu_thread"` or `"-lmkl_intel-thread"`

  Atlas library: `-lptf77blas –lptcblas`

  fftw libray: `-lfftw3_threads`

  On a computer using the module command, active multi-threaded feature:

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

2- LINEAR ALGEBRA
USE VENDOR MULTI-THREADED LIBRARY WITH SCALAPACK

- Choice of vendor library depends on the computer architecture
  - On **Intel**-based computers, use “Math Kernel Library” (mkl)
  - On **ARM**-based computers, use “ARM Performance libraries”
- Use customized link line
  - Add `with_linalg_libs=“-L… -l…”` in configuration file
- Activate **ScaLapack**
  - Add `with_linalg_flavor=“scalapack”` in configuration file
  - Add ScaLapack in link line
- Activate **multithreading**
  - Add threads in **link line**
    Ex.: `-lmkl_gnu_thread`

2- LINEAR ALGEBRA – CONT’D

- Use the “module” command to find the link line
  - `module avail` -> find the name of the *scalapack* or *mkl* module
  - `module show name_of_module` -> list predefined variables
- Use the predefined environment variables in configuration file
- *Example on “cobalt” computer (CCRT, French Computing Center)*

```
with_linalg_flavor=“scalapack”
with_linalg_libs=${SCALAPACK_LDFLAGS}
```

2- LINEAR ALGEBRA – CONT'D

Example on "cobalt" computer (CCRT, French Computing Center)

```
>> module show scalapack
-------------------------------------------------------------------
/opt/Modules/default/modulefiles/libraries/scalapack/mkl/17.0.0.098:

conflict    scalapack
prereq                  mkl/17.0.0.098
prereq                  mpi
module-whatis           MKL ScaLAPACK routines ILP64 Multi-threaded
Setenv SCALAPACK_ROOT   /ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl
Setenv SCALAPACK_INCDIR /ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include
Setenv SCALAPACK_LIBDIR /ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/lib/intel64

Setenv SCALAPACK_LDFLAGS -L/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/lib/intel64 -
lmkl_intel_ilp64 -lmkl_core -lmkl_intel_thread -lmkl_scalapack_ilp64 -lmkl_blacs_openmpi_ilp64 -lpthread
-lm

Setenv SCALAPACK_CFLAGS -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include
Setenv SCALAPACK_CXXFLAGS           -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include
Setenv SCALAPACK_FFLAGS -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include
append-path CCC_LDFLAGS -L/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/lib/intel64 -lmkl_intel_ilp64 -
lmkl_core -lmkl_intel_thread -lmkl_scalapack_ilp64 -lmkl_blacs_openmpi_ilp64 -lpthread -lm
append-path CCC_CFLAGS  -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include
append-path CCC_CXXFLAGS            -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include
append-path CCC_FFLAGS  -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include
```

2- LINEAR ALGEBRA – CONT'D

Strongly recommended

Use of ELPA library

Eigenvalue SoLvers for Petaflop-Applications

Fritz-Haber-Institut der Max-Planck-Gesellschaft

rzg

TECHNISCHE UNIVERSITÄT MÜNCHEN

Max-Planck-Institut für Mathematik in den Naturwissenschaften

BERGISCHE UNIVERSITÄT WUPPERTAL

IBM

- Usually available on supercomputers

- Needs *ScaLapack*

- Add it in linear algebra flavor

```
with_linalg_flavor="scalapack+elpa"
```

- Add include files and library:

```
with_linalg_incs="-I${ELPA_INCDIR}"
with_linalg_libs="${SCALAPACK_LDFLAGS} -L${ELPA_LIBDIR} –lelpa"
```

3- FAST FOURIER TRANSFORM
USE FFTW WITH MULTI-THREADING ACTIVATED

- FFTW is an open-source library implementing FFT
  It includes parallel FFT using MPI and multithreaded FFT

  Intel architecture : FFTW is included in the MKL library

- **Activate FFTW** in configuration file
  Add `with_fft_flavor="fftw3"`

- Use customized **link line**; activate **multithreaded** version

```
with_fft_incs="-Ifftw_path_include"
with_fft_libs="-Ifftw_path_lib -lfftw3_threads -lfftw3 -lfftw3f"
```

3- FAST FOURIER TRANSFORM – CONT'D

Example on "cobalt" computer (CCRT, French Computing Center)

Use FFTW included in MKL

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

4- PLUGINS (FALLBACKS)
USE PRE-INSTALLED VERSION IF POSSIBLE

- **netCDF/netCDF-fortran** is always present on a supercomputer

  ```
  module load netcdf
  ```
  or
  ```
  module load netcdf-fortran
  ```

- There is possibly no internet connection on a supercomputer:
  Download the plugins(fallbacks) tar file(s) before compiling
  and put them in `$HOME/.abinit/tarballs` directory

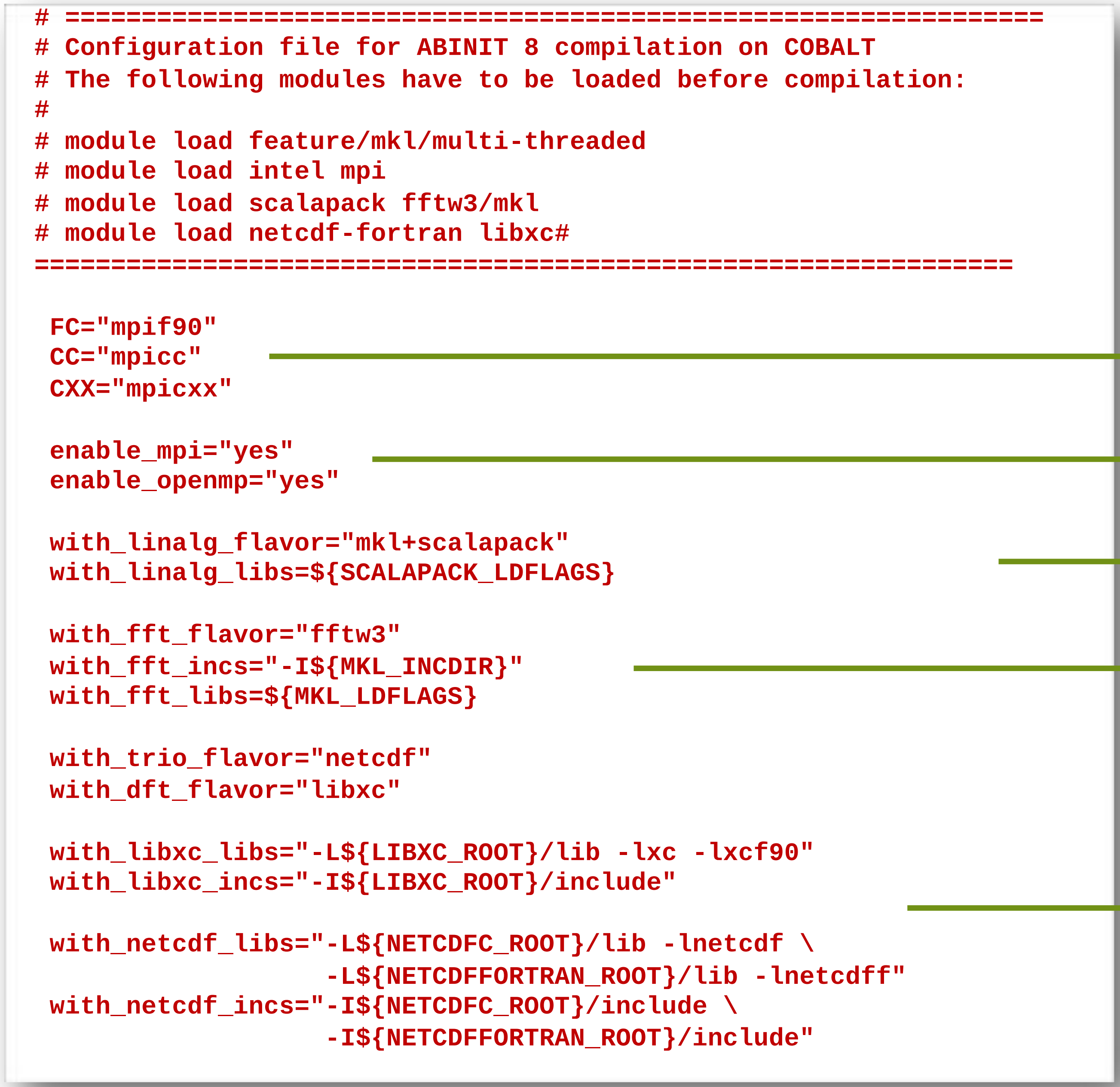
- Use customized **link line** for the pre-installed plugins

  ```
  with_netcdf_libs="-L${NETCDF_ROOT}/lib –lnetcdf -lnetcdff”
  with_netcdf_incs="-I${NETCDF_ROOT}/include"

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

FINAL CONFIGURATION FILE

Example on "cobalt" computer (CCRT, French Computing Center)



```
# ==================================================================
# Configuration file for ABINIT 8 compilation on COBALT
# The following modules have to be loaded before compilation:
#
# module load feature/mkl/multi-threaded
# module load intel mpi
# module load scalapack fftw3/mkl
# module load netcdf-fortran libxc#
==================================================================

 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"
```

→ Choice of compilers

→ Hybrid parallelism

→ MKL ScaLapack

→ FFT from MKL

→ Pre-installed plugins (netcdf, libXC)

FINAL COMPILATION REPORT

Example on “cobalt” computer (CCRT, French Computing Center)

```
===========================================================
==== Final remarks                                      ===
===========================================================

Summary of important options:

  * C compiler       : intel version 17.0
  * Fortran compiler: intel version 17.0
  * architecture     : intel xeon (64 bits)

  * debugging        : basic
  * optimizations    : standard

  * OpenMP enabled   : yes (collapse: yes)
  * MPI    enabled   : yes
  * MPI-IO enabled   : auto
  * GPU    enabled   : no (flavor: none)

  * TRIO   flavor = netcdf
  * TIMER  flavor = abinit (libs: ignored)
  * LINALG flavor = mkl+scalapack (libs: auto-detected)
  * ALGO   flavor = none (libs: ignored)
  * FFT    flavor = fftw3 (libs: user-defined)
  * MATH   flavor = none (libs: ignored)
  * DFT    flavor = libxc

Configuration complete.
```

CONCLUSION

INSTALLING ABINIT – KEYS POINTS

- **Configuration** of the build is the critical point
  Look at final report of the configuration
  Use a configuration file: `name_of_computer.ac`

- On scalar architecture
  - Activate at least **netCDF** and **libXC** plugins
  - Build a parallel executable (MPI)
  - Use preinstalled BLALS/Lapack libraries

- On parallel architecture
  - Use “module” command
  - Activate **hybrid parallelism** (MPI+openMP)
  - Link to **vendor libraries**;
    use **multithreaded** libraries

INSTALLING ABINIT – RESSOURCES

- `config.log` file, if error during configuration
- A lot of configuration file examples in
  `~abinit/doc/build/config-examples`
- https://forum.abinit.org
- https://wiki.abinit.org
- Some videos on YouTube (search for “abinit install”)