

FROM RESEARCH TO INDUSTRY



[www.cea.fr](http://www.cea.fr)

ABINIT Hands-on 2017

*A newcomer-oriented school to ab initio nanoscience simulations*

January 30-February 3, 2017 - Bruyères-le-Châtel, France

# INSTALLING ABINIT

LAPTOPS – WORKSTATIONS - SUPERCOMPUTERS

*Marc Torrent*

*CEA, DAM, DIF, France*

A terminal window with a menu bar (File, Edit, View, Search, Terminal, Help) and a list of configuration options for ABINIT. The options are: enable\_mpi="yes", enable\_mpi\_io="yes", with\_mpi\_prefix="/usr", with\_netcdf\_incs="-I/usr/include", with\_netcdf\_libs="-L/usr/lib -lnetcdf -lnetcdf", 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 -lcbblas -latlas", with\_dft\_flavor="@tempaw+bigdft+libxc+wannier90", and enable\_gw\_dpc="yes".

```
File Edit View Search Terminal Help
enable_mpi="yes"
enable_mpi_io="yes"
with_mpi_prefix="/usr"
with_netcdf_incs="-I/usr/include"
with_netcdf_libs="-L/usr/lib -lnetcdf -lnetcdf"
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 -lcbblas -latlas"
with_dft_flavor="@tempaw+bigdft+libxc+wannier90"
enable_gw_dpc="yes"
```



## ABINIT installation - Basics

What do you need?

Optional plugins: fallbacks

Some specific computing architectures

## How to obtain an executable

Good practices

How to improve the default

Configuration file

## How to obtain an efficient executable

Activate MPI+*openMP*

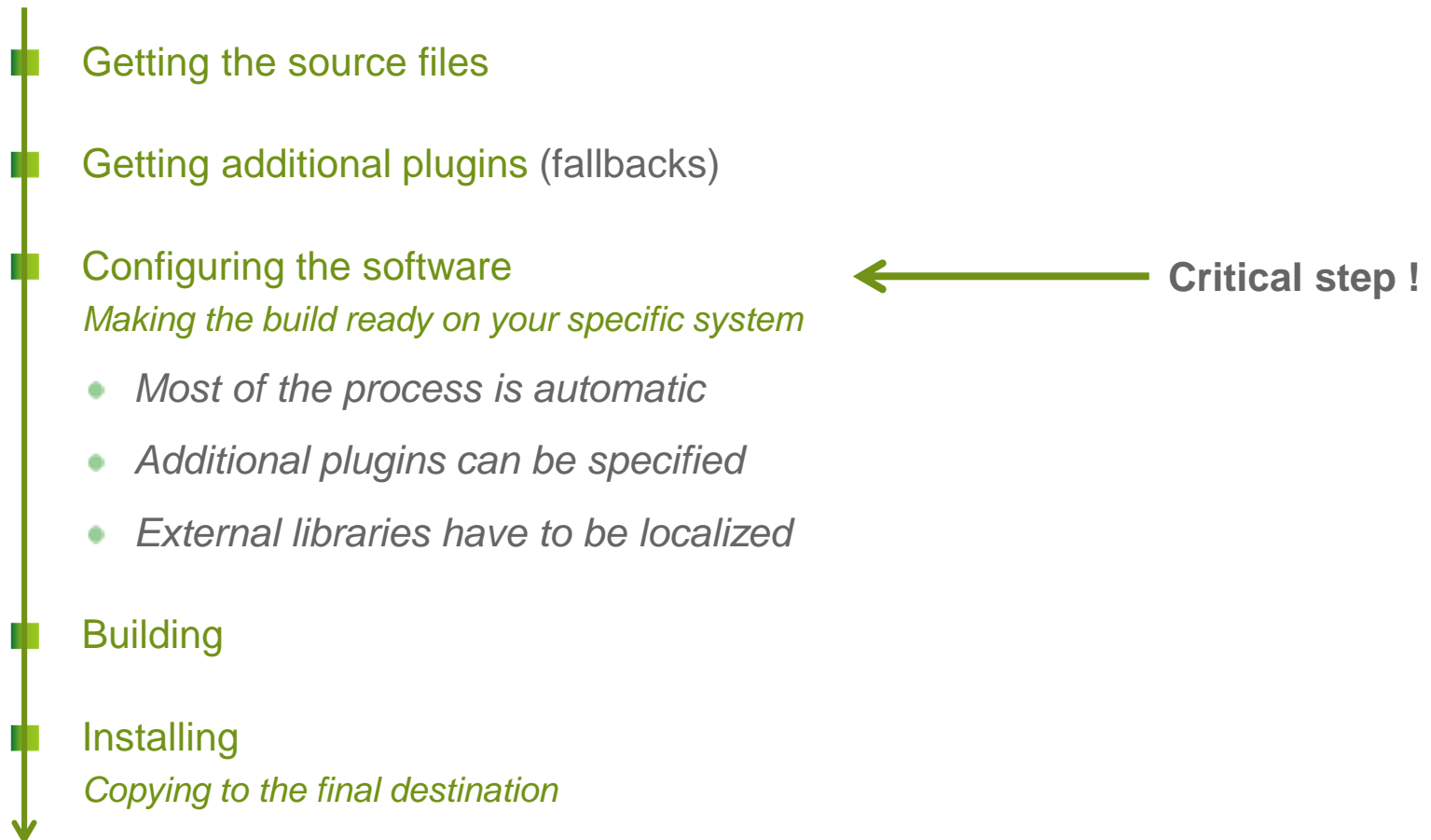
Use vendor libraries for linear algebra and FFT

# **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](http://www.abinit.org)
  
- **Internet** connection ?  
Can be convenient to download “on the fly” some extra packages

# THE COMPILATION PROCEDURE



# GETTING THE INSTALLATION FILE (TARBALL)

- The source package

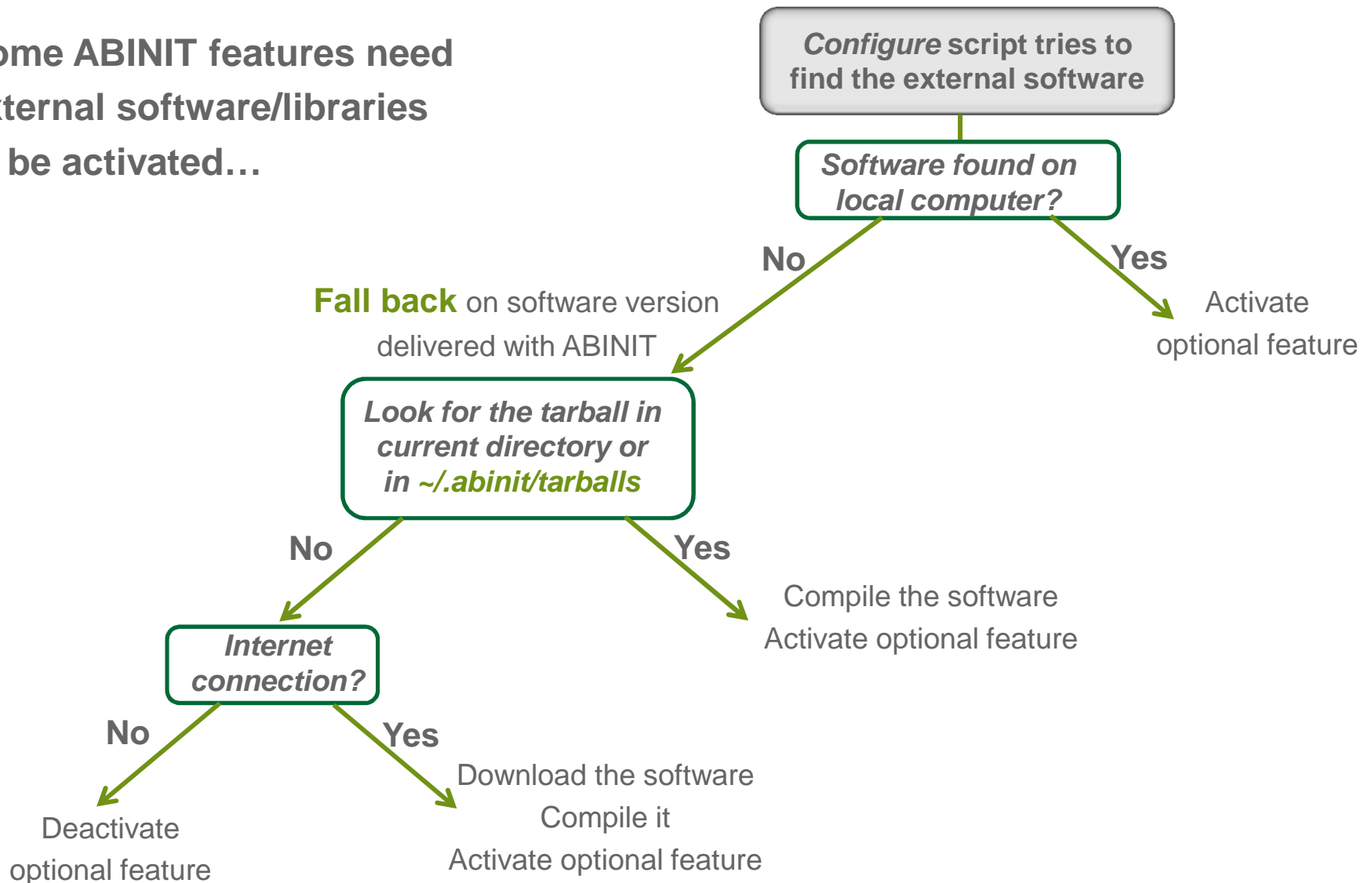


- The additional “fallbacks” (plugins)



# WHAT ARE THE FALLBACKS?

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



# 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)*
- **bigDFT** : to activate the possibility to use a wavelet basis
- **AtomPAW** : the PAW atomic data generator



```
../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

- Several executable files are copied into the destination directory :

`make install`

- **abinit** : *main executable. All-in-one software : DFT, DFPT, DMFT, MBPT, PIMD, NEB, ...*
- **cut3d** : *post-processing tool : extracting data, convert ABINIT output files into common data format*
- **anaddb** : *mandatory in the case of response function calculation ; ANALyse the Derivative DataBase*
- **macroav** : *macroscopic average technique applied on potentials*
- **conducti** : *transport properties (conductivity, reflectivity, linear optics, ...)*
- **aim** : *Bader Atom-in-Molecule analysis*

## SPECIFIC ARCHITECTURE : LINUX DISTRIBUTION UBUNTU, REDHAD, ...

- 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.

## ■ The simplest and fastest method: using homebrew package manager

- Open a terminal
- Install `homebrew`  
Everything explained here: <http://brew.sh>
- Install `abinit`: `brew install homebrew/science/abinit`
- Only libXC and netCDF fallbacks activated

## ■ Automatic but longer method: using macports package manager

- Install macport  
See <http://www.macports.org>
- Install `abinit`: `sudo port install abinit`
- Can take a long time

## ■ Manual method: compile by yourself

- 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 preload.

# 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]
```

**This step is the  
most important**

■ 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
```

# 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 or a PC but you can do better
- 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=/usr/bin/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.

## ■ 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.

## ■ First step : be sure to build a parallel executable

```
configure --enable_mpi
```

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`

In some cases, we have to localize MPI manually

```
configure --enable-mpi FC=mpif90 \  
--with-mpi-incs="-Ipath_to_incs"  
--with-mpi-libs="-Lpath_to_libs -lmpi.."
```

## ■ 2<sup>nd</sup> 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
* ALGO flavor = none (libs: ignored)
* FFT flavor = none (libs: ignored)
* MATH flavor = none (libs: ignored)
* DFT flavor = libxc
```

Configuration complete.

- **2<sup>nd</sup> 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?

## ■ 3<sup>rd</sup> 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
```

TRIO=Transferable Input Output  
netCDF

DFT plugins= LibXC, atomPAW, etc.

Configuration complete.

## ■ 3<sup>rd</sup> 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

Suppress "- -"  
Replace "-" by "\_"

```
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"
```

HOW TO OBTAIN  
AN **EFFICIENT** ABINIT EXECUTABLE

PARALLEL COMPUTERS

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

```
make multi multi_nprocs=16
```
- **Load the “modules”** (module load ...)  
Before the compilation  
Before the execution
- Efficiency on supercomputers implies:
  - use of **hybrid parallelism** (*MPI+openMP*)
  - use of preinstalled **vendor libraries** (*linear algebra, FFT*)
  - use of **parallel** version of **libraries**

# 1- ACTIVATE OPENMP (MULTITHREAD)

- Add:  
`--enable-openmp` or `enable_openmp="yes"`
- Activate “multithreaded” versions of libraries

Examples:

Intel mkl library: `“-lmkl_gnu_thread”` or `“-lmkl_intel-thread”`

Atlas library: `-lptf77blas -lptcblas`

fftw library: `-lfftw3_threads`

## 2- LINEAR ALGEBRA

### USE VENDOR MULTITHREADED 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
```



## Strongly recommended Use of ELPA library

- 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"
```

### Eigenvalue Solvers for Petaflop-Applications



Fritz-Haber-Institut  
der  
Max-Planck-Gesellschaft



Max-Planck-Institut  
für Mathematik  
in den Naturwissenschaften



## 3- FAST FOURIER TRANSFORM USE FFTW WITH MULTITHREADING 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)

```
>> module show fftw3
```

*Use fftw included in MKL*

```
-----  
/opt/Modules/default/modulefiles/libraries/fftw3/mkl/17.0.0.098:
```

```
conflict    fftw3
prereq      mkl/17.0.0.098
module-whatis  MKL FFTW3 routines
Setenv FFTW3_ROOT    /ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl
Setenv FFTW3_INCDIR  /ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include/fftw
Setenv FFTW3_CFLAGS  -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include/fftw
Setenv FFTW3_CXXFLAGS -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include/fftw
Setenv FFTW3_FFLAGS  -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include/fftw
append-path CCC_CFLAGS -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include/fftw
append-path CCC_CXXFLAGS -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include/fftw
append-path CCC_FFLAGS -I/ccc/products/mkl-17.0.0.098/default/17.0.0.098/mkl/include/fftw
```

Software description:

```
Name      : MKL FFTW3 routines
Description : Intel Math Kernel Library FFTW3 routines
Version    : 17.0.0.098
Keywords   : library mathematics
```

```
with_fft_flavor="fftw3"
with_fft_incs="-I${FFTW3_INCDIR}
with_fft_libs=${MKL_LDFLAGS}
```

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

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

```
module load netcdf
```

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

## FINAL CONFIGURATION FILE

```
# =====
# Configuration file for ABINIT 8 compilation on COBALT
# The following modules have to be loaded before compilation:
# module load scalapack
# module load fftw3
# module load elpa
# module load netcdf
# module load libxc
# =====
```

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

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

Choice of compilers

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

Hybrid parallelism

```
with_linalg_flavor="mkl+scalapack+elpa"
with_linalg_libs="${SCALAPACK_LDFLAGS} -L${ELPA_LIBDIR}" -lelpa"
with_linalg_incs="-I${ELPA_INCDIR}"
```

MKL ScaLapack +ELPA

```
with_fft_flavor="fftw3"
with_fft_incs="-I${FFTW3_INCDIR}"
with_fft_libs=${MKL_LDFLAGS}
```

FFT from MKL

```
with_trio_flavor="netcdf"
with_dft_flavor="libxc"
```

Pre-installed plugins  
(netcdf, libXC)

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

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

# 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+elpa (libs: user-defined)
* ALGO           flavor = none (libs: ignored)
* FFT            flavor = fftw3 (libs: ignored)
* 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
  - Uink to pre-installed plugins if possible
  - Link to ELPA if possible



# 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”)



Commissariat à l'énergie atomique et aux énergies alternatives

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019