The build system of Abinit 6: on the road again

Yann Pouillon

Nano-Bio Spectroscopy Group
Facultad de Químicas, Universidad del País Vasco UPV/EHU
Donostia-San Sebastián, Spain

Han-sur-Lesse, Belgium — 2011/04/11
Summary of previous episodes

Abinit 5 $\rightarrow$ Abinit 6

“Do not let what you are stay in the way of what you want to become.”

- Break backward-compatibility with Abinit 4
- Modularize build system (subsystems)
- Create build-system test suite
- Properly document build-system UI
- Take packagers into account
Build-system test suite

- Condition: all tests runnable before the build

- 01–09: check status of source tree
- 10–19: basic build-system consistency
- 20–29: enforce good maintenance practices

01: Bazaar conflict markers
02: source line lengths
03: one include of config.h in each source file
10: consistency of configuration
   \[\rightarrow\] environment.conf + options.conf + build-config.ac
11: consistency of build examples (test farm)
12: consistency of CPP options
20: forbidden flags in build examples (test farm)
   \[\rightarrow\] e.g. no optimizations in FCFLAGS_EXTRA
Current issues

TODO

1. Slow down source package size growth (Fatboy Abinit™)
2. Accept more external non-integrable dependencies
3. Handle more complex plugins (libraries, binaries, data)
4. Permit more complete use of plugin capabilities
5. Improve detection of external dependencies for packaging
6. Integrate continuously with test farm
7. Address evolving programming practices & requirements
From plugins to connectors/fallbacks

**Plugins**
- flat structure (manual interdependency checking)
- poor detection of external dependencies
- preemptive interpretation of user requests
- non-transparent overriding of options
- optimized for home-built Abinit versions

**Connectors/Fallbacks**
- hierarchical structure (automatic interdependency checking)
- advanced detection of external dependencies
- strict respect of user requests
- rigorous connector → fallback → error behavior
- can seamlessly handle home-built and packaged versions
From plugins to connectors/fallbacks

- **Plugins**
  - flat structure (manual interdependency checking)
  - poor detection of external dependencies
  - preemptive interpretation of user requests
  - non-transparent overriding of options
  - optimized for home-built Abinit versions

- **Connectors/Fallbacks**
  - hierarchical structure (automatic interdependency checking)
  - advanced detection of external dependencies
  - strict respect of user requests
  - rigorous connector $\rightarrow$ fallback $\rightarrow$ error behavior
  - can seamlessly handle home-built and packaged versions
Structure of the connectors (Abinit 6.6)
Flavors of the connectors (Abinit 6.6)

<table>
<thead>
<tr>
<th>Connector</th>
<th>Flavor</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI</td>
<td><em>with_mpi_prefix</em> or CC=..., CXX=..., FC=...</td>
</tr>
<tr>
<td>GPU</td>
<td><em>cuda-single</em>, cuda-double</td>
</tr>
<tr>
<td>TIMER</td>
<td><em>abinit</em>, gptl, none, papi</td>
</tr>
<tr>
<td>TRIO</td>
<td><em>etsf_io</em>, fox, <em>netcdf</em></td>
</tr>
<tr>
<td>FFT</td>
<td>asl, custom, fftw2, fftw2-threads, fftw3, fftw3-mkl, fftw3-threads, <em>none</em>, sgimath</td>
</tr>
<tr>
<td>MATH</td>
<td>gsl, mlib, <em>none</em></td>
</tr>
<tr>
<td>DFT</td>
<td><em>atompaw</em>, <em>bigdft</em>, <em>libxc</em>, <em>wannier90</em></td>
</tr>
</tbody>
</table>

- ScaLAPACK: goto-mpi, mkl-mpi, netlib-mpi
- Custom flavors: bypass build-system checks
- TRIO, DFT: additive flavors (e.g. “etsf_io+fox”)
Connector-related options (Abinit 6.6)

- Philosophy: priority on precision
- 2 kinds of flavors: exclusive or additive
  - `--with-*-flavor="(<value>|custom|none)"
    - ex: `--with-fft-flavor="fftw3"
    - ex: `--with-trio-flavor="etsf_io+fox+netcdf"
    - ex: `--with-dft-flavor="atompaw+bigdft+libxc+wannier90"
  - `--with-*-bins="..."
  - `--with-*-incs="..."
  - `--with-*-libs="..."
- `--enable-connectors: for maintenance
- `--enable-fallbacks: force external libs if disabled
- See `~abinit/doc/config/build-config.ac for details
Structure of the fallbacks

- Grouping: package-wise $\rightarrow$ task-wise
  (uncompress, patch, config, build, install, clean)

- Refactored subtree
  - makefiles
  - patches
  - sources
  - install

- One include path, one library path, one binary path
- Simplified & fully automatic cleaning procedure
- Can select versions to build
- Profiles (e.g. abinit-6.4, abinit-6.6)
- Can be detached from Abinit & used by other projects
Modularization of the build system

Objective

Allow teams to work asynchronously on different Abinit components
⇒ independent fine-tuning of different blocks
⇒ easier debugging of individual components

- Different parts ←→ different tasks
- Core + bindings: set build parameters
- Fallbacks: delegate builds
- Test suite: set runtime parameters
- Documentation: set publishing parameters
- Data: ensure completeness and consistency
Modularization of the build system

Objective

Allow teams to work asynchronously on different Abinit components

⇒ independent fine-tuning of different blocks

⇒ easier debugging of individual components

- Different parts ↔ different tasks
- Core + bindings: set build parameters
- Fallbacks: delegate builds
- Test suite: set runtime parameters
- Documentation: set publishing parameters
- Data: ensure completeness and consistency
Coming soon ...

- New freeze of the build-system UI
  ➞ submit feature requests **NOW!**

- Comprehensive documentation of the build-system UI
  ➞ improved ease of use

- Activation of subsystems
  ➞ asynchronous fine-grained tuning

- Complete rewrite of the build-system internals
  ➞ optimization for speed & efficiency

- Libtool support
  ➞ export of dynamic shared objects
Acknowledgments

- Whole Abinit community

- Special thanks: Xavier, Jean-Michel, Alain, Damien, Marc, Matthieu, Josef

- Build-system crash test award: Matteo Giantomassi
Special requests

Service evaluation: acknowledgments in papers

« We thank Yann Pouillon for valuable help/support/whatever... »
Send a copy of final paper to mailto:yann.pouillon@ehu.es

Looking for a position

Research engineer profile: research + services
Service part focused on software (hardware < 10%)
Starting before Spring 2012 if possible
Forward proposals to mailto:yann.pouillon@ehu.es
Service evaluation: acknowledgments in papers

"We thank Yann Pouillon for valuable help/support/whatever..."
Send a copy of final paper to mailto:yann.pouillon@ehu.es

Looking for a position

Research engineer profile: research + services
Service part focused on software (hardware < 10%)
Starting before Spring 2012 if possible
Forward proposals to mailto:yann.pouillon@ehu.es