Building and Running EC-Earth 3

A short guide

Table of Contents

Introduction

EC-Earth[1] is a global coupled climate model, which integrates a number of component models in order to simulate the earth system. It is developed by a consortium of European research institutions, which collaborate in the development of a new Earth System Model. The goal of EC-Earth is to build a fully coupled Atmosphere-Ocean-Land-Biosphere model, usable from seasonal to decadal climate prediction and climate projections.

EC-Earth 3 is (as of 2012) the most recent version of that system and the one described here. The intention of this guide is to document mostly technical aspects that are of importance when getting started with EC-Earth 3.

In order to gather the latest and most relevant information about EC-Earth 3 development in one place, a Development Portal has been established. This is a Web-based service, which can be reached via the following URL:

```
https://dev.ec-earth.org
```

The site includes further documentation, issue tracking facilities, and access to the version control system. It is strongly recommended to refer to the Development Portal first when any concerns or requests need to be addressed.

Preparations

Technical Prerequisites

A number of tools and libraries are needed to configure and build EC-Earth 3:

Prerequisite	Notes
Compiler/Languages	
GNU make	GNU make version 3.81 or more is needed to build IFS
Fortran	A Fortran 77/90/95 compliant compiler with preprocessing capabilities is needed.
C/C preprocessor	
Python	The half-automatic build configuration tool ec-conf is tested with Python v2.4.3 and above. It will, however, not work with Python v3.0+
Bash	Used by FCM
Perl	Used by FCM
Libraries	
MPI	
BLAS/LAPACK	
NetCDF	
GRIB API	Needed for IFS GRIB I/O. Tested with version 1.9.9. Download from

1) W. Hazeleger et al (2010) *EC-Earth: a seamless Earth system prediction approach in action.* Bull Am Meterol Soc 91:1357–1363. doi:10.1175/2010BAMS2877.1

Prerequisite	Notes
	http://www.ecmwf.int/products/data/software/grib_api.html
GRIBEX	Needed for IFS GRIB I/O. Tested with version 370.
Other tools	
makedepf90[2]	Needed, if dependency files for the IFS code need to be created (because they where removed or invalidated by a change of source files). This tool is provided both pre-compiled (for Linux) and as source code in the EC-Earth package.
TkInter (Python module)	Needed if the graphical interface of the ec-conf tool is used.
GNU date (64-bit version)	Needed for full run-script functionality.

Getting the Sources

EC-Earth 3 is distributed in source form, hence, it must be configured for the platform in use and built (compiled and linked) by the user. The principal distribution method for the source code is to access the Subversion (SVN) server, which is part of the EC-Earth 3 Development Portal. The SVN server is located at the URL

```
https://svn.ec-earth.org
```

In order to understand the SVN repository structure in more detail, please refer to the Wiki article at https://dev.ec-earth.org/projects/ecearth3/wiki/EC-EARTH_3_Version_control_strategy. The following instructions assume that the most recent status in the development of EC-Earth 3 (known as trunk) is to be accessed. For other purposes, e.g. for accessing an older version, the URL's have to be adjusted accordingly.

There is a certain freedom in how to lay out the source code and runtime directories in the user's file system. Therefore, the following instructions may be modified, provided that the changes are consistent.

The following sequence of commands provides a complete structure of the EC-Earth 3 source, runtime, and documentation directories:

```
> svn checkout --username USER_NAME --password PASSWORD \
           https://svn.ec-earth.org/ecearth3/trunk/sources ECEARTH3_BASE_DIR
> cd ECEARTH3_BASE_DIR
> svn checkout --username USER_NAME --password PASSWORD \
           https://svn.ec-earth.org/ecearth3/trunk/runtime
> svn checkout --username USER_NAME --password PASSWORD \
           https://svn.ec-earth.org/ecearth3/trunk/doc
```

Note that the highlighted terms have to be replaced by the Development Portal account details (USER_NAME, PASSWORD) and a user chosen directory name (ECEARTH3_BASE_DIR), which will hold all of the EC-Earth 3 components. Note also that a backslash (\) has been used to indicate that the line has been split in this guide for better readability.

For the remainder of this guide, it will be assumed that the source code was ob-

2) Available at http://personal.inet.fi/private/erikedelmann/makedepf90 under the GNU General Public License version 2.

tained by the above procedure, thus, the directory structure is expected to follow the mentioned layout.

Please refer to the documentation of the SVN client that is being used about how the user name and password information is handled. It might be possible to store the account details, thus avoiding specifying them repeatedly. However, be aware of security implications.

Building

Build Configuration

Once the source code of EC-Earth 3 is in place, the systems needs to be configured for building. The recommended method for build configuration is to use the ec-conf tool, in fact, this is the only method described in this guide. Manual configuration is, however, possible by editing the ec-conf configuration files by hand, albeit being an error-prone procedure.

The ec-conf tool is provided in the `ECEARTH3_BASE_DIR/util/ec-conf` sub-directory and it comes with it's own documentation in `ECEARTH3_BASE_DIR/doc`. The usage of ec-conf is only very briefly explained here, as far as it is needed to understand the EC-Earth 3 build configuration process.

All configurable parameters are collectively stored in an XML data base file, which is to be adapted by the user according to the computing platform and build environment. For a detailed description of the XML file syntax, please refer to the ec-conf user guide. Once the XML data base file has been adapted, it is subsequently used by the ec-conf tool to create the required configuration files for all components. Hence, from a user's perspective, a *single place, single syntax configuration* is achieved.

Summarising, the usage of ec-conf comprises two stages, namely

- Adapting the configuration parameter in the XML data base file, and
- Running the ec-conf tool to create the actual configuration files.

Any editor that handles plain-text files can be used during the first step. Thereafter, the second step facilitates the command line interface of ec-conf. Alternatively, both steps can be dealt with in one go by utilising the ec-conf graphical user interface (GUI) by using the `--gui` option. Please refer to the ec-conf user guide for a detailed description of entire process.

When the sources are obtained according to the instructions in the previous chapter, and XML data base file is provided at

```
ECEARTH3_BASE_DIR/config-build.xml
```

as a basis for build configuration. This file needs to be complemented with a `Platform` section suitable for the user's computing platform and the configuration parameters must be adjusted for the particular environment. It is strongly recommended to add new `Platform` sections when needed rather than re-using existing ones. This is to make sure that platform-dependent configurations are persistently documented.

Assuming that the XML data base file has been adjusted, the ec-conf command line interface is invoked as

```
> ec-conf --platform <PLATFORM> config-build.xml
```

provided that the ec-conf command is present in the search path. The <PLATFORM> argument refers to the corresponding Platform section in the XML file. Upon completion of this command, all configuration files are created in their proper places. It is important to repeat that step every time the configuration is changed. Note that ec-conf may display warnings about empty parameter values. As long as the corresponding parameters are deliberately left without values (e.g. because a certain platform does not support or need a specific parameter), these warnings can be ignored.

Compiling

Assuming that the build configuration has been carried out with the ec-conf tool, the compilation of the component models is the next step. As both IFS and NEMO depend on OASIS, the coupler software has to be compiled first. Afterwards, the order of building the remaining components is not important.

Compiling Oasis

Provided that ECEARTH3_BASE_DIR is the current directory, the build process is started from the build directory of OASIS:

```
> cd oasis3
> cd util/make_dir
```

where compilation of the libraries and the executable is initiated by the command

```
> make BUILD_ARCH=ecconf -f TopMakefileOasis3

--> Using BUILD_ARCH=ecconf
--> Reading configuration from

[... more output follows ...]

make[1]: Leaving directory ...
```

Upon successful completion of the process (i.e. when no errors are displayed), the OASIS libraries, module files, and the executable can be found in the ECEARTH3_BASE_DIR/oasis3/ecconf directory.

Compiling NEMO

From NEMO version 3.3 onwards, the Flexible Configuration Management (FCM)[3] system is used for compilation. Hence, the way NEMO is compiled has changed since earlier EC-Earth releases. The FCM configuration file has been adapted to support ec-conf and it is recommended to use this tool to build NEMO within EC-Earth 3.

The main script for compiling NEMO is located in the CONFIG subdirectory

```
> cd nemo-3.3.1
> cd CONFIG
```

and it is used like this:

```
> ./makenemo -n ORCA1L46_LIM3 -m ecconf -j <NUM_PROC>
```

By applying the above command line, it is assumed that ec-conf has been successfully run and that the default configuration (comprising the ORCA1 ocean grid with 46 vertical levels and the LIM3 sea ice model) is to be built. Other configurations are

3) http://cms.ncas.ac.uk/index.php/fcm

available, which can be listed (among other information) by

```
> ./makenemo -h
Usage  : makenemo [-h] [-n name] [-m arch] [-d dir1 dir2] [-r conf] [-j No]
[... more output follows ...]
Available configurations :
ORCA1L46_LIM2   [...]
ORCA1L46_LIM3   [...]
ORCA1L75_LIM2   [...]
ORCA1L75_LIM3   [...]
ORCA025L46_LIM2 [...]
ORCA025L46_LIM3 [...]
ORCA025L75_LIM2 [...]
ORCA025L75_LIM3 [...]
```

Note that the compilation of NEMO (when calling the `./makenemo` script as shown before), it is possible to specify the number of parallel compilation processes with the `-j <NUM_PROC>` command line argument.

When compilation has succeeded, a line similar to

```
Build command finished on <SOME_DATE>.
```

should appear and two executables, `nemo.exe` and `server.exe`, are located in the `ORCA1L46_LIM3/BLD/bin` subdirectory (or a similar one corresponding to the chosen configuration). Note that it is possible to compile and keep more than one configuration of NEMO at a time. In fact, this is supported by the EC-Earth 3 run scripts. Note also that, unlike earlier versions, the number of processors is no longer compiled into the NEMO executable, but given in a namelist at run-time.

Compiling IFS

The IFS source files and build scripts are located in the ifs-36r4 subdirectory of `ECEARTH3_BASE_DIR`:

```
> cd ifs-36r4
```

All makefiles have been configured by ec-conf, hence, the build step can be started immediately. As a first step, all IFS libraries are compiled. The build process supports parallel make, which is activated by the `-j` command line argument. The number of parallel compilation processes, which can be used efficiently, has to be established experimentally on the build platform, but it is probably less than ten. The make command to compile the libraries reads

```
> make BUILD_ARCH=ecconf -j <NUM_PROC> lib
```

After all libraries are built successfully, the IFS executable (the ifs-master) has to be linked. There is no advantage in using parallel make in that step as only one compilation instance is involved:

```
> make BUILD_ARCH=ecconf master
```

When the linking process has completed, the IFS executable will be found in the `bin` subdirectory.

NOTE: Due to the current structure of the makefiles, it is not possible to specify both the `lib` target and the `master` target at the same time in the make command line.

Running EC-Earth 3

Run Configuration with ec-conf

In order to run EC-Earth 3, run scripts have to be prepared to reflect the current platform and the experiment at hand. The recommended way of doing this is by using the ec-conf tool, much in the same way as when building EC-Earth 3. The ec-conf tool is provided in the ECEARTH3_BASE_DIR/util/ec-conf sub-directory. Because ec-conf has its own documentation in the ECEARTH3_BASE_DIR/doc directory only the usage of the tool, not the tool itself, is described here.

Setup Prerequisites for Using ec-conf

Preparing the run scripts with the ec-conf tool on a specific platform requires the user to provide:

- A platform dependent template file, which provides the shell functions `configure()`, `launch()`, and `finalise()`. This functions are required to handle, respectively, any platform dependent configuration, launch mechanism for an MPMD MPI job, and, if necessary, post run operations.
- All configurable parameters, e.g., platform dependent run time paths and specific experiment settings. These have to be stored in an XML data base file for the use with ec-conf.

The first step is needed only once for each platform. Existing template files are located (again assuming the earlier defined directory structure) at

```
ECEARTH3_BASE_DIR/work/<platform>.cfg.tmpl
```

and can server as templates for new platforms.

In the second step, some of the settings in the XML data base file are platform dependent (`RUN_DIR`, `PROC_PER_NODE`, ...) and some of them need to be updated according to each experiment (`EXP_NAME`, `RUN_START_DATE`, ...).

An example XML database file for the run scripts is provided at

```
ECEARTH3_BASE_DIR/work/config-run.xml
```

This file, together with the existing template files and the documentation on the XML database file structure in the separate ec-conf documentation, serves as a guide when setting up EC-Earth 3 on a new platform.

Generating Run-scripts with the ec-conf Command Line Interface

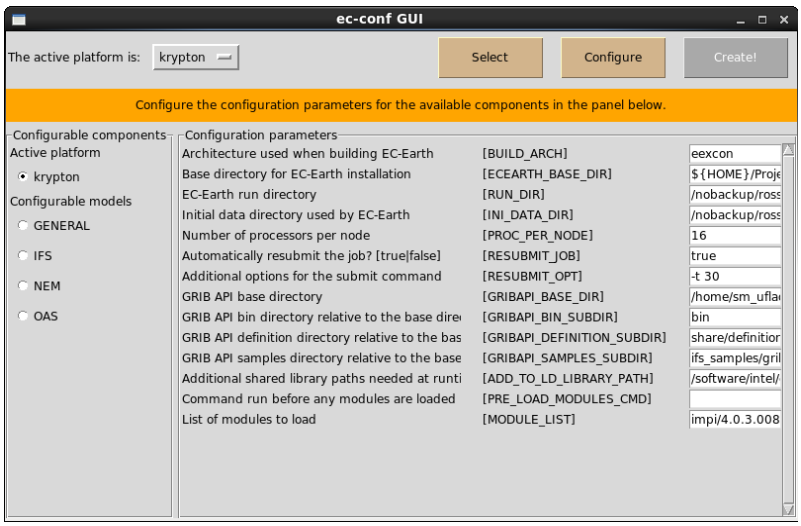
Assuming the user has a template file for a given platform and has filled out all the configurable parameters in the XML database file in his/her favourite text editor, the actual run scripts are created by issuing,

```
> ec-conf --platform <PLATFORM> <RUN_CONFIG_FILE>
```

As when building EC-Earth 3, <PLATFORM> refers to the appropriate platform section in the XML file in use and <RUN_CONFIG_FILE> is the XML database file. This command will create two scripts,

- `run.sh` Run-script for a coupled experiment
- `run-atm.sh` Run-script for an atmosphere-only (IFS) experiment

Generating run-scripts with the ec-conf Graphical User Interface



The ec-conf GUI while being used for run-script generation

Alternatively, the run-scripts can be prepared with the Graphical User Interface (GUI) of ec-conf, which provides all of the functionality of the Command Line Interface. In fact, the GUI might turn out to be preferable when used to create run-scripts in a more interactive way. It is, for example, possible to generate a number of run-scripts, which differ in in just a few, specific aspects, such as the start date of the experiments. Thus, the run-scripts can later be used in a batch-like fashion without the need for manually copying and modifications.

The ec-conf GUI is launched with the command

```
> ec-conf --gui <RUN_CONFIG_FILE>
```

The ec-conf GUI also allows the user to interactively update the XML database file, save it for future use and generate the run-scripts. See the ec-conf documentation for details on the GUI usage.

General ec-conf Hints

Some general advice for using ec-conf include:

- Make sure ec-conf is in your search path
- If you are using relative paths in the <RUN_CONFIG_FILE>, make sure they are consistent with your current working directory used when running ec-conf
- ec-conf warnings are prefixed *WW*, and output to `stderr`. Check if the warning is relevant to your particular case, if not, it can be ignored
- ec-conf errors are prefixed *EE* and cannot be ignored as no run-scripts are generated

Running EC-Earth 3 Experiments

The EC-Earth 3 Wiki includes instructions about how to submit the generated run-

scripts on a number of platforms. The Wiki also points out where the default initial data files can be found for that platform. Users who are setting up EC-Earth 3 on new platforms are encouraged to add corresponding information to the Wiki.