Source Code, Compilation and Installation Notes

January 2006

Ariane 1.3.0

Bruno.Blanke@univ-brest.fr – NicolasGrima@univ-brest.fr

Table of contents

1 Introduction

Ariane is a Fortran code dedicated to the computation of 3D streamlines in a given velocity field (as the output of an Ocean General Circulation Model) and subsequent water masses analyses. In this document a new version of this application is presented. It is based on the old version written by Bruno Blanke [Speich, 1992; Blanke et Raynaud, 1997; Blanke et al., 1999 and 2001] in Fortran 77.

This new version using Fortran 90 capacities is more modular and readable. Dynamic allocation is introduced to a better memory management and array notation and intrinsic functions are used to improve portability and performances.

Compilation and installation use *autotools*[1] efficiencies. Ariane migration from a platform to another is easier and needs a minimum of effort. *Cpp keys*, inherited form OPA[2] coding rules, are removed and replaced by a *namelist* file that avoids excessive recompilations.

All the input data file strategy has been rethinked using *netcdf* facilities. For example, this new version of Ariane application reads directly data from OGCM *netcdf* output[3].

To be complete this version comes with a Users' Guide and this note to take entirely advantage of the Ariane application.

This note describes how to build and install the Ariane package, version 1.2.*x*, on Unix and Linux systems. In a first part, the Ariane official web page, package and documentation will be presented and in a second part, its compilation and installation will be detailed.

This new Ariane version has been tested intensively by its creator (B. Blanke) but keep in mind that it has emerged from its cocoon and is drying its wings. Bugs, problems and comments should be sent to Bruno.Blanke@univ-brest.fr and/or Nicolas.Grima@univ-brest.fr.

2 Official resources

2.1 Web page

Ariane package (source code), news, documentation, support and contacts can be found at:

`http://www.univ-brest.fr/lpo/ariane`

For a complete description of the Ariane product and utilities consult the Ariane Users' Guide.

[1] automake, autoconf and configure, make, etc.

[2] The OPA system is an Ocean General Circulation modelling System shared by projects (research and operational) in oceanography and Climate change studies . It is developed at the Laboratoire d'Océanographie DYnamique et de Climatologie.

[3] The only condition is that the data have to be stored on a C grid (Arakawa classification [Ref ..]).

2.2 Ariane package

The Ariane package is available in the download section of the Ariane official web pages. It is a compressed “tarball” where you can find source codes, documentation, tests and all the useful files to compile and install easily this package.
This product access is free, nevertheless we strongly recommend to subscribe to the mailing list. News, help and support will be available only for people whose email address are recorded. It is also for us the way to evaluate the popularity of this product, and to know who uses it and what for.

2.3 Documentation

The documentation is available on the Ariane official web page (html, pdf) and in the Ariane package (pdf) in the “ariane-*x.x.x*/doc/manual” directory.
It is also possible to dive into the source code using your favourite web-browser. Html pages are available in the “ariane-*x.x.x*/doc/src_browser” directory of the Ariane package.

3 Compilation and Installation

We assume that the Ariane package will be compiled and installed on a Unix, Linux or Mac OS X operating system.

3.1 A quick start

To install Ariane, uncompress and unpack the compressed tarball file, then move to the ariane-*x.x.x* directory (where *x.x.x* is the product version):

```
gunzip  ariane-x.x.x.tar.gz
tar -xf ariane-x.x.x.tar
cd ariane-x.x.x
```

Now run the usual configure, make, make check and make install cycle:

`./configure`	to configure the compilation and installation.
`Make`	to compile executables.
`make check`	to test Ariane in qualitative and quantitative mode.
`make install`	to install the distribution (bin, doc, examples).

The *configure* script will try to find necessary tools in your **`PATH`** environment variable.
Configure comes with a lot of useful options and environment variables. Enter `configure --help` to print them all.
For example, when you run *configure* you may optionally use the **`--prefix`** argument to change the default installation directory (`/usr/local`)”:

```
./configure --prefix=/my/dir
```

This Ariane version needs *netcdf* 3.6.0 (or newer) and requires the avaibility of the netcdf Fortran 90 module (*netcdf.mod*). If by default your *netcdf* version is older or doesn't support Fortran 90 calls, please install a new and complete version and use the *netcdf* environment variables **`NETCDF_INC`** and **`NETCDF_LIB`** to override default *configure* choices.
For example in csh and ksh:

```
setenv NETCDF_INC /usr/local/netcdf-3.6.0/include
setenv NETCDF_LIB /usr/local/netcdf-3.6.0/lib
```

or

```
export NETCDF_INC=/usr/local/netcdf-3.6.0/include
export NETCDF_LIB=/usr/local/netcdf-3.6.0/lib
```

Where you have to change `/usr/local/netcdf-3.6.0` into the correct path for your platform.

If all this doesn't work, then you might have to read the next chapter. Better luck next time!

3.2 Requirements

3.2.1 Autotools, configure and make concepts

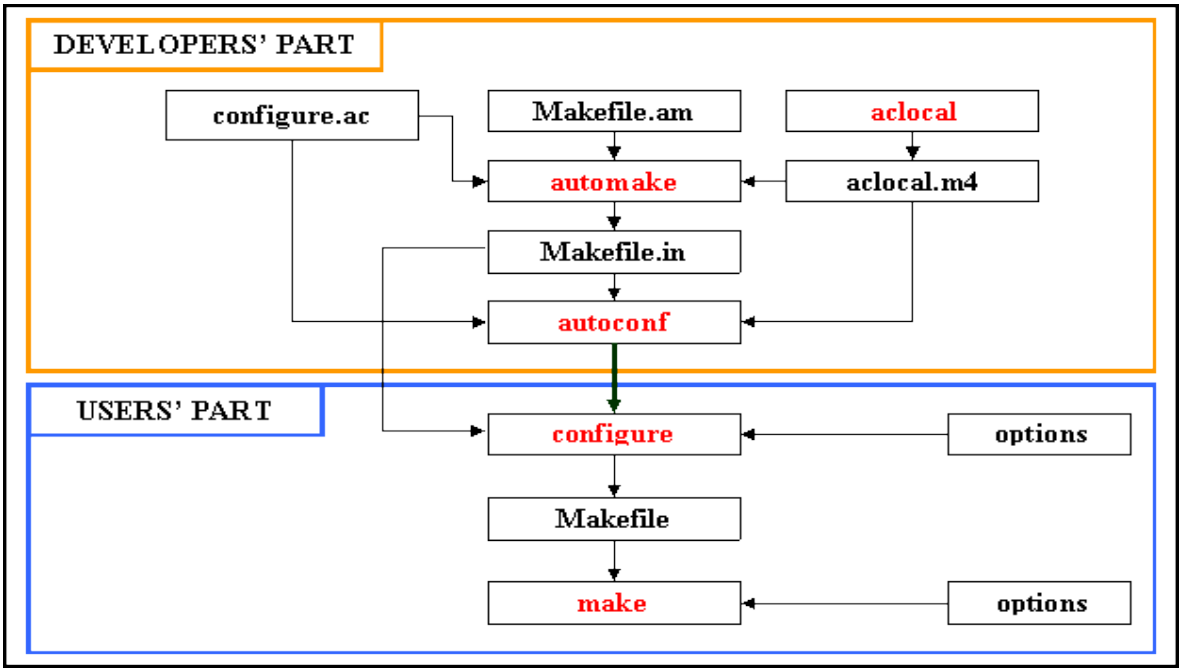

Figure 1: development and compilation scheme based on autotools.

To develop and maintain the Ariane package we use *autoconf* and *automake* (*autotools-Figure 1*) which provide an effective build system to maintain the application and companion tools.
Automake is a tool for generating *Makefile*'s--descriptions of what to build-- that conform to a number of standards. *Automake* substantially simplifies the process of describing the

organization of a package and performs additional functions such as dependency tracking between source files.

Autoconf is a tool that makes your packages more portable by performing tests to discover system characteristics before the package is compiled. Your source code can then adapt to these differences.

Practically, the purpose of the *autotools* is to create the platform independent input files necessary for the *configure* script to run correctly. These *autotools* are used by the developers and maintainers and not by the Ariane users. On the other hand the *configure* script is designed to be run by the user installing the Ariane package on his own platform.

Running the *configure* command on the build host executes the various tests originally specified by the *configure.ac* file, and then creates another script, *config.status*. This new script generates *Makefile*'s from the named *Makefile.in*'s. Once *config.status* has been created, it can be executed by itself to regenerate files without rerunning all the tests.

The final tool to be run is *make* with specific options to compile and install the package. Like *configure*, it is designed to execute on the build host. *make* will use the rules in the generated *Makefile* to compile the project sources with the aid of various other scripts generated early on.

All this to conclude that *configure* and *make* (or *gmake*) commands have to be installed and have to run successfully on your platform to compile and install without problem the Ariane package.

3.2.2 Fortran 90/95 compiler

A Fortran 90/95 compiler is required to generate the Ariane application and its tools.

The *configure.ac* file takes a no-exhaustive list of Fortran 90/95 compilers into account. This list is presented in Table 1:

Compiler command	Compiler full name and provider	Platform/O.S.	Tested
f90	It's a generic name for a lot of Fortran 90/95 compilers.	HP alphaserver SGI NEC SX	Yes Yes Yes
g95	GNU Fortran 90/95 compiler	PC/Linux Mac (G4)/Mac OS X	Yes Yes
ifort	Intel Fortran compiler	PC/Linux	Yes
pgf90	Portland Group Fortran compiler	PC/Linux	No
pathf90	Pathscale Fortran 90 compiler	PC(AMD)/Linux	No
xlf90/xlf95	IBM Fortran compiler	Power 4 or 5/AIX	No
ifc	Intel Fortran compiler (older version of ifort)	PC/Linux	No

Table 1: Fortran 90/95 compiler taking into account by the Ariane configure script.

If your Fortran 90/95 compiler is not present in this list, it is always possible to use *configure* environment variables to force the compilation with it. In this case, use **FC**, **FCFLAGS** and/or

LDFLAGS to specify, Fortran 90/95 compiler name, options and link editor for your platform respectively. These environment variables override the choices made by *configure* or help it to find libraries and programs with non-standard names/locations. They are detailed in chapter 3.2.1.
If no Fortran 90/95 compiler seems to be installed on your machine, verify your environment variable PATH and see with your administrator system if there are no troubleshooting. As a last resort install a Fortran 90/95 compiler (for example: g95).
In all cases, please feel free to send us your comment and suggestions to share them in the future version of the package.
To close this chapter, we want to specify that due to the short time and CPU resources need to compile and install this package, cross compilation is not available.

3.2.3 NetCDF library

NetCDF (Network Common Data Form) is an interface for array-oriented data access and a library that provides an implementation of the interface. The *netcdf* library also defines a machine-independent format for representing scientific data. Together, the interface, library, and format support the creation, access, and sharing of scientific data.
More information is available on the official web site:

`http://www.unidata.ucar.edu/packages/netcdf/`

All the input data file strategy of the Ariane package has been rethinked using *netcdf* facilities. The **3.6.0** (or a newer) **version** of the *netcdf* package, including the Fortran 90 interface, has to be installed on you computer to compile and link successfully the application. If this version is not the default version on your computer, it is possible to specify a different directory with environment variables during the "configuring" phase (see chapter 3.3.1).
Ariane application can read input data from different *netcdf* files which have not necessarily the same number of fields and the same number of time steps. Therefore it is possible to read directly data from OGCM *netcdf* output (if these data are on a C grid in the Arakawa classification [Arakawa, 1972]) without duplicating them in a different format or in a specific file. The NetCDF file name strategy is explained in the Ariane Users' Guide.
Unfortunately in this version the possibility to start in the first NetCDF file at a time step different from 1 is not implemented. It is not possible also to reduce extraction to a specific region, but all these restrictions should be removed in the next version of the application.

3.3 Compilation and installation detailed

Ariane package is developed to be used on multiple platforms. Since each of these platforms have different compilers and different include files, there is a need to write *Makefile*'s and build scripts so that they can work on a variety of platforms.
As presented in the requirement chapter, *autotools* are chosen to solve these installation problems. If you have downloaded and built any GNU software from source, you are probably familiar with the *configure* script and the *make* command.

The *configure* script runs a series of tests to determine information about your machine and the *make* command, using *Makefile*'s features, compiles, installs, etc the application.

3.3.1 Configure

A standard configuration of the compilation and installation requires simply the execution of the *configure* script as follow:

```
./configure
```

This script will determine your system environment and will try to find and test the default Fortran 90 compiler and *netcdf* library. By default it will assume that the installation will be made in the `/usr/local` directory.

The configure scripts support a wide variety of options. The most interesting ones are `--prefix` and `--enable`. You can use the `--help` option to get a list of the options controlling the Ariane compilation and installation:

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

Configure Script Options		
--help	To display the configure help and exits.	
	./configure --help	
--prefix	To overwrite the product install directory which is by default /usr/local.	
	./configure --prefix=/my/home/login/product	
--enable-optimization=*level*	To select the compiling optimization level [normal].	
	normal	a soft Fortran 90 optimization level is selected for the robustness in results.
	debug	debugging Fortran 90 options are selected to debug the application.
	aggressive	an aggressive Fortran 90 optimization level is selected for performances. In this case, results must be verified.
	./configure –-enable-optimization=aggressive	
--enable-profiling	To add profiling Fortran 90 options if they are available for this compiler. By default the `gprof` profiler is used to analyze Ariane code performances.	
	./configure –-enable-profiling	
Environment Variables		
FC	To specify a Fortran 90 compiler.	
	export FC=f90 (ksh) and ./configure [options]	

NETCDF_INC	To specify the include NetCDF directory.
	export NETCDF_INC=/dir/netcdf_3.6.x/include (ksh) and ./configure [options]
NETCDF_LIB	To specify the library NetCDF directory.
	export NETCDF_LIB=/dir/netcdf_3.6.x/lib (ksh) and ./configure [options]

Table 2: configure options and environment variables.

Use the options and environment variables listed in Table 2 to override the choices made by *configure* or to help it find libraries and programs with non-standard names/locations.
It is possible to use them together and environment variables must be declared before to submitting the *configure* script.
At the end of the *configure* execution a *config.log* file is created where all the *configure* execution is detailed and it is useful to consult it if an error is encourtered afterwards.

3.3.2 Make

`Make` is a command that creates a machine-language program by compiling source files to produce a group of object files, and then to link the object files together. It will use the rules of the generated *Makefile* to do this with the aid of various other scripts generated earlier. If modifications are made in the source code, it recompiles only the modules that have been updated since the last compilation.

Running *make* (or *gmake*) will build the Ariane application (`ariane`) and utilities (for example: `mkseg` and `mkseg0`).
Run *make* like this:

`make` (or `gmake`)

It is possible to add a key-word to pass a specific action to the *make* command. This key-word is a target that defines a dependency rule in the *Makefile.*
Use the target `clean` to delete all the object files and executables generated by the `make` command:

`make clean`

3.3.3 Make check

`Make check` tests on small cases the executable built by the *make* command. In our case *make check* will test the Ariane application. It will submit a first test in quantitative mode and a second test in qualitative mode (more information about these modes is available in the Users' Guide).
Tests should be finished respectively by one of these messages:

```
PASS: quant_check.sh          PASS: quali_check.sh
==================            ==================
```

All 1 tests passed
==========

All 1 tests passed
==========

3.3.4 Make install

To install the entire Ariane package, run the installation like this:

```
make install
```

Files are automatically distributed in the appropriate places as shown in Figure 2: executables (bin), documentation (doc) and examples (examples).

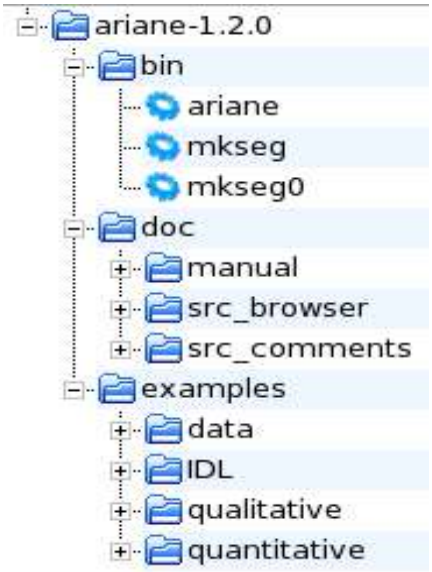

Figure 2: Installation tree.

Add the *bin* subdirectory of the installation directory to your path:

```
setenv PATH prefix/bin:$PATH        (csh)
export PATH=prefix/bin:$PATH        (ksh)
```

Where *prefix* is the product install directory defined during the configuration phase (see chapter 3.3.1).

Check that everything is in order at this point by typing

```
which ariane        (csh, tcsh, etc)
type ariane         (bash, sh, ksh, etc)
```

Here is a table with different *Makefile* targets and their definitions.

Targets	Definitions
make install	To install the entire Ariane package: executables, documentation, examples, scripts and so on.
make install-exec	To install only the executables in the bin directory.

`make install-data`	To install part of the Ariane package: documentation and examples but not the executables.
`make uninstall`	To delete all the installed files, the copies that the install target creates. This rule should not modify the directories where compilation is done, only the directories where files are installed. Directories are not deleted.

Table 3: make keywords.

4 References

- Arakawa, A., 1972: Design of the UCLA general circulation model. Numerical simulation of weather and climate. Dept. of Meteorology, University of California, Rep. 7, 1-34.
- Blanke, B., and S. Raynaud, 1997: Kinematics of the Pacific Equatorial Undercurrent: a Eulerian and Lagrangian approach from GCM results. J. Phys. Oceanogr., 27, 1038-1053.
- Blanke, B., M. Arhan, G. Madec, and S. Roche, 1999: Warm water paths in the equatorial Atlantic as diagnosed with a general circulation model. J. Phys. Oceanogr., 29, 2753-2768.
- Blanke, B., S. Speich, G. Madec, and K. Döös, 2001: A global diagnostic of interocean mass transfers. J. Phys. Oceanogr., 31, 1623-1632.
- Speich, S., 1992: Étude du forçage de la circulation océanique par les détroits: cas de la Mer d'Alboran. Thèse de l'Université Pierre et Marie Curie, Paris, France.