Installation and Administration of the PELICANS Platform

March 15, 2010

INSTITUT DE RADIOPROTECTION ET DE SÛRETÉ NUCLÉAIRE
DIRECTION DE PRÉVENTION DES ACCIDENTS MAJEURS
SERVICE D'ETUDES ET DE MODELISATION DES INCENDIES, DU CONFINEMENT ET DU CORIUM
Centre d'Études de Cadarache - Bâtiment 702 - BP3-13115 SAINT PAUL LEZ DURANCE CEDEX
Téléphone : (33) 4.42.19.96.48 - Fax : (33) 4.42.19.91.66 - E-mail : bruno.piar@irsn.fr

Key Notions

Apart from describing the installation and administration tasks of the PELICANS platform, this report should clarify various notions that are listed below in the jargon of PELICANS.

- *installation directory*
- *environment configuration*
- *administration makefile*
- *compilation makefile*
- *compiler architecture*
- *extra-makefile*
- *architecture-makefile*
- *external API*

The notion of *compilation level* is mentioned many times. It is explained in details in [1].

Documentation

The **whole** documentation of PELICANS is available
at: **`doc/Site/index.html`**
(in the *installation directory*)

Chapter I

Installation of PELICANS

The installation of PELICANS on a UNIX (*eg* LINUX, Solaris, Mac OS X) system is described here.

I.1 License Acceptance

- The installation of PELICANS proceeds on the basis of a Java ARchive (Java 1.5 or higher required), also called a *JAR* file.
- The *JAR* file can be freely downloaded. Its name has the following form:

  pelicans_*version***-installer.jar**

  where *version* is a number identifying the current release.
- At the beginning of the installation procedure, which is described below, the license agreement will be fully displayed and acceptance notifications will be requested to ensure that any user of PELICANS has knowledge of the CeCILL-C license and accepts its terms.

I.2 Unpacking the JAR File

- In the following, we consider the example of the JAR file **pelicans_3.0.0-installer.jar** corresponding to the **3.0.0** release.
- The JAR file may be unpacked by the command:

  ```
  java -jar pelicans_3.0.0-installer.jar
  ```
- Accepting the terms of the license, and choosing an appropriate installation directory, *eg* **/usr/local/**, leads to the directory structure of figure I.1, where **/usr/local/pelicans-3.0.0/** is called the *installation directory*. It will be the current/working directory until the end of this chapter.
- There is no need to understand the details of the directory structure of figure I.1.

  Nevertheless three points are essential.

  1. The file **./doc/Site/index.html** is the entry point for the whole documentation (including the present report).
  2. The file **Makefile** is devoted to the platform administration (§I.3).
  3. The files **./bin/init.csh** and **./bin/init.sh**, that remain to be built (§I.3.2), are devoted to the *environment configuration* before using PELICANS (§I.3.2 and [1]).

```
/usr/local/pelicans-3.0.0
              ├── include
              ├── ExternalAPI
              ├── FrameFE
              ├── Geometry
              ├── LibraryApplications
              ├── LinearAlgebra
              ├── PDEsolver
              ├── PELbase
              ├── RefSol
              ├── SingleApplication
              ├── StandardApplications
              ├── UnitTests
              ├── admin
              ├── bin
              ├── doc ──────── Site ····· index.html
              ├── etc
              ├── include
              ├── tools
              └···························· Makefile
```

FIGURE I.1: directory structure after unpacking the tarball.

I.3 Administration Makefile

The file called **Makefile**, located in the *installation directory* specifies all the necessary instructions for the administration tasks of the PELICANS platform. It is called the *administration makefile*.

Using the *administration makefile* requires:

- GNU **make** version 3.77 or newer;
- **perl** version 5.6 or newer.

Invoking **make** without any argument in the *installation directory* leads to a simplified help:

```
/usr/local/pelicans-3.0.0/ 3>  make
Usage: make CCC=<arg> <target>
<target>
    environment : build environment configuration files
    all         : build libraries
    check       : check install of built-in functionalities
    check_<API> : check activation of an External API with
                  <API> = MPI, PETSc, UMFPACK, METIS, Aztec
                          or SPARSKIT
    clean       : remove generated files
    maintainer-help : help for advanced maintainers
<arg> : compiler system (default: gcc)
```

PELICANS is written in **C++98** (ISO/IEC 14882:1998) and is an open-source project, which means that the *source code* (the human-readable form in which it was developed) is open for anyone to see and modify. In fact, PELICANS is only distributed as source code by IRSN. Before it can be used for constructing applications, it must be transformed into low-level machine code instructions (unreadable by most humans) gathered in files called libraries. That process is called *compiling* (or *building* because it usually involves many different steps) and is performed by a specific set of programs

collectively called *a compiler*.

A **C++98** compiler is needed to install PELICANS. The compiler name, that will be denoted hereafter symbolically **<compiler>**, must be assigned to the **CCC** variable[1] of the *administration makefile* by invoking **make** with any of the targets as follows:

```
make CCC=<compiler> <target>
```

The default for **<compiler>** is **gcc**. Hence:

```
make <target>
```

is equivalent to:

```
make CCC=gcc <target>
```

I.3.1 Main Targets of the Administration Makefile

The four main targets of the *administration makefile* are given below.

- **environment** : build the files devoted to the environment configuration that is required before using PELICANS.
- **all** : make first the target **environment** then build the PELICANS libraries.
- **check** : build the executable required by the tests of the built-in functionalities, then run these tests.
- **clean** : delete all the files that may have been built when making the targets **environment**, **all** and **check**.

Strictly speaking, the command:

```
/usr/local/pelicans-3.0.0/ 4>  make CCC=<compiler> all
```

is sufficient to produce a usable version of PELICANS.

Nevertheless a more complete understanding of the platform administration tasks may be valuable.

I.3.2 Environment Configuration

The target **environment** builds the configuration files **init.csh** and **init.sh** in the **./bin** directory:

```
/usr/local/pelicans-3.0.0/ 4>  make environment
      ...  license agreement  ...
Construction of bin/init.csh
Construction of bin/init.sh
****
* target environment completed
****************************************
```

Sourcing **bin/init.csh** on **csh** type UNIX shells, or **init.sh** on **sh** type UNIX shells, is required to configure the environment expected by PELICANS:

- the environment variable **PELICANSHOME** is set to the proper value;
- the management utility **pel** is accessible.

For instance, in the case of a **csh** type UNIX shell:

[1] the exact meaning of the value assigned to **CCC** will be given in §II.2.2.

```
/usr/local/pelicans-3.0.0/ 5>  source bin/init.csh
/usr/local/pelicans-3.0.0/ 7>  echo $PELICANSHOME
/usr/local/pelicans-3.0.0
/usr/local/pelicans-3.0.0/ 8>  which pel
/usr/local/pelicans-3.0.0/bin/pel
```

At this point, the detected *compiler architecture* must be checked.

I.3.3 Check of the Compiler Architecture

The notion of *compiler architecture* will be exposed in chapter II. As of now, the command:

```
/usr/local/pelicans-3.0.0/ 9>  pel arch -v <compiler>
```

should **not** end with a message of the type:

```
FAILED: unable to find an Architecture-Makefile.
```

However, if such is the case, the creation of a valid *compiler architecture* is required (§II.5) before continuing. Otherwise, the command **pel arch** returns the name of the *compiler architecture* (a character sequence without any endline) that will be symbolically denoted **<arch>** in the following. For example:

```
/usr/local/pelicans-3.0.0 10>  pel arch gcc ; echo
Linux-gcc
```

in which case **<arch>** ≡ **Linux-gcc**.

I.3.4 Build of the PELICANS Libraries

The command:

```
/usr/local/pelicans-3.0.0/ 11>  make CCC=<compiler> all
```

builds the PELICANS libraries. More precisely:

- a directory **./lib/<arch>** is created;
- two *compilation makefiles* are created (by the command **pel depend** [1]):
  ./lib/<arch>/opt0/Makefile
  ./lib/<arch>/opt1/Makefile
  corresponding respectively to the two *compilation levels* **opt0** and **opt1** [1];
- two PELICANS libraries are built (by the command **pel build**), corresponding to the two *compilation levels* **opt0** and **opt1**. They are files with basename **libpel0** and **libpel1**, located in **./lib/<arch>**.

At this point, the installation of PELICANS is complete.

I.3.5 Check of the Installation

▷ The command:

```
/usr/local/pelicans-3.0.0/ 12>  make CCC=<compiler> check
```

performs the actions described below.

- A directory **./tests** is created.
- The PELICANS-based application defined by the sources located in the **./StandardApplications** directory is built with the **opt0** compilation level.
  The resulting executable is **./tests/lib/<arch>/opt0/exe** .

- The above executable is run with the data file **./admin/check.pel** to perform a predefined subset of the tests located in subdirectories of **./UnitTests**, **./StandardApplications**, **./LibraryApplications**.
- Similarly, the special PELICANS-based application defined by the sources located in the **./SingleApplication** directory (example of main program overriding) is built with the **opt0** compilation level.
  The resulting executable (**./tests/SingleApplication/<arch>/opt0/exe**) is used to perform the tests whose data deck is located in subdirectories of **./SingleApplication** .

The results of these actions should be interpreted as follows.

- The fact that all tests are "successful" means that the checking procedure revealed no error in the installation of PELICANS.
- The occurence of a "failing test" means that the installation of PELICANS cannot be considered successful.
- The occurence of an "ambiguous" test may not signify the failure of the installation. For example, since the application devoted to the tests compares the runs just performed with the reference ones that may have been obtained on a different machine, slight numerical differences may be acceptable.

In this latter case, a more thorough analysis of the complete test results, located in the **./tests/<arch>** directory, may be necessary to (in)validate the installation of PELICANS.

▷ The **check** target of the administration makefile, described above, only adresses the built-in functionnalities provided by PELICANS. The enabling of the chosen *external APIs* must be tested using specific targets.

I.3.6 Sharing the Installation Directory

The *installation directory* may be accessed by computers of different nature (both in hardware and software) *via* a network. It is called a *shared resource*.

The mechanism of *compiler architecture* facilitates this sharing by isolating compilation and testing results in particularized directories. Hence the steps of §I.3.3, §I.3.4, §I.3.5 may be repeated as many times as there are different *compiler architectures* interacting with the *installation directory*.

Finally, new files and directories have been added to those already represented in figure I.1. They are sketched in figure I.2.

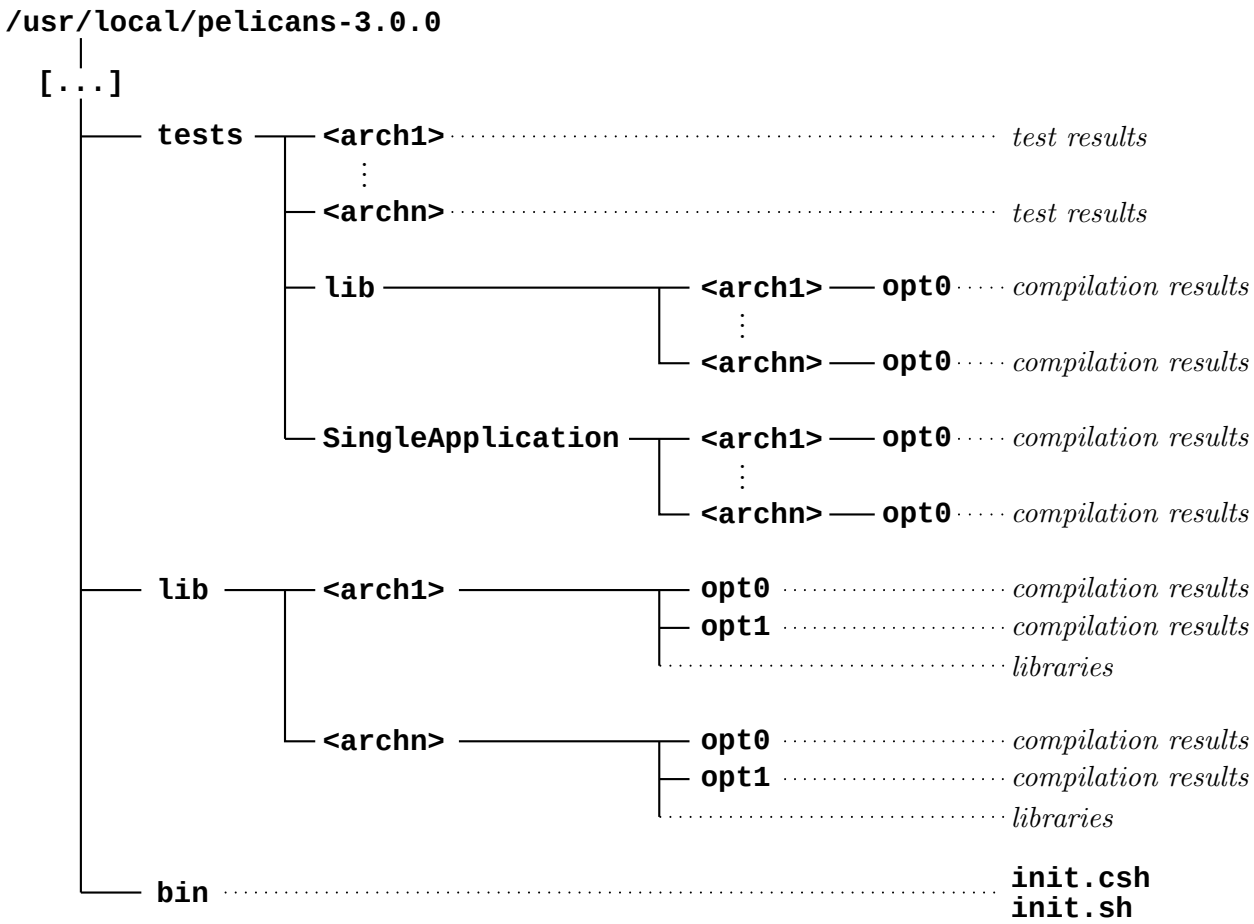

FIGURE I.2: sketch of the new files and directories added to the *installation directory* when it is accessed in the context of *n compiler architectures* **<arch1>** ... **<archn>**.

Chapter II

Compiler Architectures and Compilation Makefiles

II.1 Introduction

PELICANS is delivered with an *administration makefile* that controls all the steps for building the platform. It defines all the necessary commands (and their dependencies) for managing PELICANS libraries, the associated documentation, the tests and the satellite tools.

The same kind of *administration makefile* should be used for managing all PELICANS-based applications. As an example, on may consider **`$PELICANSHOME/etc/Makefile_for_appli`** .

For compiling tasks, these *administration makefiles* rely on *compilation makefiles* that are generated and executed with **`pel`** commands :

$$\boxed{\text{administration } \texttt{\textbf{Makefile}}} \xrightarrow[\text{execute with } \texttt{\textbf{pel build}}]{\text{generate with } \texttt{\textbf{pel depend}}} \boxed{\text{compilation } \texttt{\textbf{Makefile}}}$$

The compilation makefile is made of four pieces, two of them being selected in accordance with the software and hardware environment. These two pieces, called respectively *architecture-makefile* and *extra-makefile* are originally stored in separate files with extension **`.mak`** that are the core of notion of *compiler architecture*.

II.2 Compiler Architecture

▷ The *compiler architecture* depends on:

- the hardware environment together with its operating system, collectively called *machine*;
- the *compiler*;
- the enabled *external APIs*, which themselves depend on the *machine* and the *compiler*.

▷ The *compiler architecture* is defined by a pair of files with extension **`.mak`** :

- the *architecture-makefile*, which essentially formalizes the usage of the current *compiler* on the current *machine*;
- the *extra-makefile*, which essentially describes the linkage of the enabled *external APIs* with PELICANS on the current *machine* with the current *compiler*.

▷ The *compiler architecture* has a name which is derived from the names of its *architecture-makefile* and its *extra-makefile*.

II.2.1 Usefulness

PELICANS uses *compiler architectures* to:

- set the options specific to the compiler that is used;
- set the compiler options specific to the hardware environment;
- set the compiler options specific to the operating system;
- enable/disable external APIs;
- isolate compilation results in particularized directories;
- isolate testing results in particularized directories.

Hence *compiler architectures* facilitate working in environments that rely on disk sharing between multiple computers with possibly different hardware and operating systems.

II.2.2 Discovery

The *compiler architecture* is found out by selecting the *architecture-makefile* and the *extra-makefile* (in *ad hoc* directories, see below) that are the more closely related to the current *machine*, the chosen *compiler* and *external APIs*. The responsibility of this task is assigned to the **`pel arch`** command.

Pattern Matching

The **`pel arch`** command searches successively two files:

1. the *architecture-makefile* called: **`<xxx>.mak`** ;
2. the *extra-makefile* called: **`extra-<xxx>.mak`** .

In both cases, **`<xxx>`** denotes symbolically a character sequence which matches one of the following patterns tried out in sequence:

checked pattern for **`<xxx>`**	example
1. [hostname]-[compiler]	**`sand-gcc`**
2. [hostname]	**`sand`**
3. [sysname]_[release]-[compiler]	**`SunOS_5.8-gcc`**
4. [sysname]-[compiler]	**`SunOS-gcc`**
5. [compiler]	**`gcc`**
6. [sysname]_[release]	**`SunOS_5.8`**
7. [sysname]	**`SunOS`**

most specific

least specific

where:

- [hostname] is the name of the current host (it may be substituted if a file called **`arch_file.cfg`** exists in the searched paths, see below);
- [sysname] is the name of the current operating system given by the UNIX command **`uname`**;
- [release] is the release of the current operating system given by the UNIX command **`uname`**;
- [compiler] is the (only) argument to the **`pel arch`** command (if any) and **`gcc`** otherwise (hence it is not necessarily the name of the command that carry out compilation).

The value of **`<xxx>`** may be (and often is) different for the *architecture-makefile* and for the *extra-makefile*. In both cases, it is the first of the checked patterns for which a valid file has been found.

Within the *administration makefile*, or within **$PELICANSHOME/etc/Makefile_for_appli**, the argument of **pel arch** is the value given to **CCC**. Hence the command (in the context of §I.3):

```
/usr/local/pelicans-3.0.0/ 11>  make CCC=toto all
```

will lead to searching a *compiler architecture* with [compiler] ≡ **toto** .

Searched Paths

First, **pel arch** searches in the directory given by the environment variable **PELARCHDIR** (if defined), then in the **$PELICANSHOME/etc** directory (if it is not defined, the subdirectory **etc** of the current directory is searched instead).

Hostname Substitution

When a file called **arch_file.cfg** is encountered in the searched paths, **pel arch** tries to substitute the current hostname by an alias name found in this file. The first match found is picked out. When no match is found, the current hostname is used.

This file is a two columns file. Comments starts with **#** (sharp). Each line describes a possible substitution:

- the first column contains a **perl** regular expression matching hostnames.
- the second column contains the alias for the regular expression.

Example of **arch_file.cfg**:

```
sinux1 pinux      # host 'sinux1' uses config called 'pinux'
sinux\d+ sinux    # other sinux hosts use config called 'sinux'
pinux\d+ pinux    # all  pinux hosts use config called 'pinux'
```

II.2.3 Name

▷ Each *compiler architecture* determined by **pel arch** receives a name. Such a name has been symbolically denoted **<arch>** or **<arch1>**...**<archn>** in the previous chapter (§I.3) . It has many practical uses, for instance to isolate compilation and testing results in particularized directories.

▷ The name of the *compiler architecture* is inferred from that of the *architecture-makefile* and the *extra-makefile*. It is split in two parts, separated by the sign **-**.

- The left part essentially identifies the *machine*. If the hostname has been substituted by an alias, it will always be that alias. Otherwise, within the two pattern matching for the discovery of the *architecture-makefile* and the *extra-makefile*, it will be the most specific among [hostname], [sysname]_[release], [sysname] if any or **undef** if none.
- The right part will be [compiler] within the two pattern matching for the discovery of the *architecture-makefile* and the *extra-makefile*, if any, or **undef** if none.

II.2.4 Examples

The **-v** option of **pel arch** is very useful to monitor the discovery of the *compiler architecture*. Let us consider the following example performed on a host of name **b08u0004.local** under Mac OS X:

```
/home/martin 1> pel arch -v
Hostname: b08u0004.local
Compiler: gcc
Architecture-Makefile searched in:
   /Users/piar/SYNC/PELICANS/DEV/etc
      b08u0004.local-gcc
      b08u0004.local-undef /usr/local/pelicans-3.0.0/etc/b08u0004.local.mak
      Darwin_9.7.0-gcc
   *  Darwin-gcc   /usr/local/pelicans-3.0.0/etc/Darwin-gcc.mak
   *  undef-gcc    /usr/local/pelicans-3.0.0/etc/gcc.mak
      Darwin_9.7.0-undef
      Darwin-undef
Extra-Makefile searched in:
   /Users/piar/SYNC/PELICANS/DEV/etc
      b08u0004.local-gcc
      b08u0004.local-undef
      Darwin_9.7.0-gcc /usr/local/pelicans-3.0.0/etc/extra-Darwin_9.7.0-gcc.mak
      Darwin-gcc
      undef-gcc
      Darwin_9.7.0-undef
   *  Darwin-undef /usr/local/pelicans-3.0.0/etc/extra-Darwin.mak
Compiler Architecture name: Darwin-gcc
```

Two files have been searched in the subdirectory **etc** of the *installation directory* **/usr/local/pelicans-3.0.0/** (which means that the environment variable **PELARCHDIR** is not defined). The ***** characters indicate the files whose name matches the checked pattern.

The *compiler architecture* is: **Darwin-gcc** .

The *compilation makefiles* will be built (see below) on basis of:

- the *architecture-makefile*: **/usr/local/pelicans-3.0.0/etc/Darwin-gcc.mak** ;
- the *extra-makefile*: **/usr/local/pelicans-3.0.0/etc/extra-Darwin.mak** .

Now, on the same machine, let us find out if a *compiler architecture* using compiler CC is available.

```
/home/martin 2> pel arch -v CC
Hostname: b08u0004.local
Compiler: CC
Architecture-Makefile searched in:
   /usr/local/pelicans-3.0.0/etc
      b08u0004.local-CC
      b08u0004.local-undef
      Darwin_9.7.0-CC
      Darwin-CC
      undef-CC
      Darwin_9.7.0-undef
      Darwin-undef
FAILED: unable to find an Architecture-Makefile.
```

Unsurprisingly the answer is no. Using this CC compiler would require the prior construction of one of the searched files.

II.3 Compilation Makefile

A *compilation makefile* is associated to

- a *compilation level*;
- a *compiler architecture*.

It is a stand-alone file specifying instructions fully compatible with GNU **`make`** version 3.77 or newer (it may be used as such, independently of the **`pel`** commands).

Compilation makefiles are assembled by the **`pel depend`** command. Their structure is briefly described below.

a compilation makefile is made of 4 pieces	options and classes – produced par **`pel depend`** ▪ definition of the directory storing the generated files (**`BINDIR`**) ▪ setting of compiler flags relative to the *compilation level* ▪ definition of the list of source files (**`SRC`**) ▪ definition of directories of include files (**`INC`**)
	extra-makefile – deduced from the *compiler architecture* ▪ status for each external API: enabled/disabled ▪ configuration of each external API
	architecture-makefile – deduced from the *compiler architecture* ▪ definition of the options specific to the compiler ▪ definition of the options specific to the machine
	generic makefile – **`$PELICANSHOME/etc/generic_targets.mak`** ▪ definition of the standard targets ▪ definition of the standard production rules (objets, dependencies)

The two pieces *extra-makefile* and *architecture-makefile* are straight inlining of the files with extension **`.mak`** that are specific to the current *compiler architecture*.

II.4 Enabling/Disabling External APIs

In the PELICANS jargon, *external APIs* denominate libraries

- that have been developed by others,
- whose license is compatible with the CeCILL-C license of PELICANS,
- that are made available to PELICANS clients through wrapper classes whose source is located in the **`$PELICANSHOME/ExternalAPI`** directory.

Examples are the numerical algebra toolkits PETSc and UMFPack.

External APIs may be used on request, provided that:

- they have been installed from another source;
- they are enabled in the *extra-makefiles* used for assembling the *compilation makefile*.

In *extra-makefiles*, what concerns an *external API*, say **`PKG`**, appears as follows:

```
WITH_PKG = 1                          enable the PKG external API
...                                   ... a little further in the file ...
ifeq ($(WITH_PKG),1)                  if PKG is enabled

ifeq ($(MAKE_PEL),1)                  if PELICANS classes are taken into account
SRC += $(wildcard $(PELICANSHOME)/ExternalAPI/Pkg/src/*.cc)
CPPFLAGS += -I$(PELICANSHOME)/ExternalAPI/Pkg/include
endif

PKGPATH = /usr/local/...              location of PKG in the file system
CPPFLAGS += -I$(PKGPATH)/include      location of PKG include files
LIBPATH  += $(PKGPATH)/lib            location of PKG libraries
LDLIBS   += -lpkg                     link with PKG library

WITH_SUBPKG = 1                       enable the SUBPKG external API required by PKG
endif
```

There are different possibilities for enabling/disabling an *external API* called **PKG**

1. In the *extra-makefile*:

- disabling :
  assign the value **0** to variable **WITH_PKG** : **WITH_PKG=0**
- enabling :
  assign the value **1** to variable **WITH_PKG** : **WITH_PKG=1**
  set the appropriate compilation and localization variables
- permanent scope :
  all *compiler architecture* based on this *extra-makefile* will be affected

2. Through the **PELBUILD** environment variable, used by **pel depend** :

- disabling :
  modify **PELBUILD** by an instruction like: **PELBUILD="$PELBUILD -with=PKG"**
- enabling :
  modify **PELBUILD** by an instruction like: **PELBUILD="$PELBUILD -without=PKG"**
- local scope :
  all *compiler architecture* within the current shell session will be affected

3. Directly inside the *compilation makefile*:

- disabling :
  as in **1.**
- enabling :
  as in **1.**
- temporary scope :
  the configuration will last until a new generation of this *compilation makefile*

II.5 Creation of a New Compiler Architecture

A *compiler architecture* may be created or modified to adapt PELICANS to particular *machines* or *compilers*.

The best way to do it is as follows.

- Copy to a special directory the nearest *extra-makefile* and/or *architecture-makefile* available in **`$PELICANSHOME/etc`** .
- Rename those files so that they can be recognized by **`pel arch`** as specific to new *compiler architecture*.
- Adapt the contents of the renamed files.
- Set up the **`PELARCHDIR`** environment variable so that it identifies the special directory.
- Check the taking into account of the new *compiler architecture* with the **`pel arch -v`** command.

Finally, PELICANS must be rebuilt (§I.3.4, §I.3.5) with respect to the new *compiler architecture*.

License for PELICANS

CeCILL-C FREE SOFTWARE LICENSE AGREEMENT

Notice

This Agreement is a Free Software license agreement that is the result of discussions between its authors in order to ensure compliance with the two main principles guiding its drafting:

- firstly, compliance with the principles governing the distribution of Free Software: access to source code, broad rights granted to users,
- secondly, the election of a governing law, French law, with which it is conformant, both as regards the law of torts and intellectual property law, and the protection that it offers to both authors and holders of the economic rights over software.

The authors of the CeCILL-C (for Ce[a] C[nrs] I[nria] L[ogiciel] L[ibre]) license are:

Commissariat à l'Energie Atomique - CEA, a public scientific, technical and industrial research establishment, having its principal place of business at 25 rue Leblanc, immeuble Le Ponant D, 75015 Paris, France.

Centre National de la Recherche Scientifique - CNRS, a public scientific and technological establishment, having its principal place of business at 3 rue Michel-Ange, 75794 Paris cedex 16, France.

Institut National de Recherche en Informatique et en Automatique - INRIA, a public scientific and technological establishment, having its principal place of business at Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay cedex, France.

Preamble

The purpose of this Free Software license agreement is to grant users the right to modify and re-use the software governed by this license.

The exercising of this right is conditional upon the obligation to make available to the community the modifications made to the source code of the software so as to contribute to its evolution.

In consideration of access to the source code and the rights to copy, modify and redistribute granted by the license, users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the successive licensors only have limited liability.

In this respect, the risks associated with loading, using, modifying and/or developing or reproducing the software by the user are brought to the user's attention, given its Free Software status, which may make it complicated to use, with the result that its use is reserved for developers and experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the suitability of the software as regards their requirements in conditions enabling the security of their systems and/or data to be ensured and, more generally, to use and operate it in the same conditions of security. This Agreement may be freely reproduced and published, provided it is not altered, and that no provisions are either added or removed herefrom.

This Agreement may apply to any or all software for which the holder of the economic rights decides to submit the use thereof to its provisions.

Article 1 - DEFINITIONS

For the purpose of this Agreement, when the following expressions commence with a capital letter, they shall have the following meaning:

Agreement: means this license agreement, and its possible subsequent versions and annexes.

Software: means the software in its Object Code and/or Source Code form and, where applicable, its documentation, "as is" when the Licensee accepts the Agreement.

Initial Software: means the Software in its Source Code and possibly its Object Code form and, where applicable, its documentation, "as is" when it is first distributed under the terms and conditions of the Agreement.

Modified Software: means the Software modified by at least one Integrated Contribution.

Source Code: means all the Software's instructions and program lines to which access is required so as to modify the Software.

Object Code: means the binary files originating from the compilation of the Source Code.

Holder: means the holder(s) of the economic rights over the Initial Software.

Licensee: means the Software user(s) having accepted the Agreement.

Contributor: means a Licensee having made at least one Integrated Contribution.

Licensor: means the Holder, or any other individual or legal entity, who distributes the Software under the Agreement.

Integrated Contribution: means any or all modifications, corrections, translations, adaptations and/or new functions integrated into the Source Code by any or all Contributors.

Related Module: means a set of sources files including their documentation that, without modification to the Source Code, enables supplementary functions or services in addition to those offered by the Software.

Derivative Software: means any combination of the Software, modified or not, and of a Related Module.

Parties: mean both the Licensee and the Licensor.

These expressions may be used both in singular and plural form.

Article 2 - PURPOSE

The purpose of the Agreement is the grant by the Licensor to the Licensee of a non-exclusive, transferable and worldwide license for the Software as set forth in Article 5 hereinafter for the whole term of the protection granted by the rights over said Software.

Article 3 - ACCEPTANCE

3.1 – The Licensee shall be deemed as having accepted the terms and conditions of this Agreement upon the occurrence of the first of the following events:

(i) loading the Software by any or all means, notably, by downloading from a remote server, or by loading from a physical medium;

(ii) the first time the Licensee exercises any of the rights granted hereunder.

3.2 – One copy of the Agreement, containing a notice relating to the characteristics of the Software, to the limited warranty, and to the fact that its use is restricted to experienced users has been provided to the Licensee prior to its acceptance as set forth in Article 3.1 hereinabove, and the Licensee hereby acknowledges that it has read and understood it.

Article 4 - EFFECTIVE DATE AND TERM

4.1 EFFECTIVE DATE – The Agreement shall become effective on the date when it is accepted by the Licensee as set forth in Article 3.1.

4.2 TERM – The Agreement shall remain in force for the entire legal term of protection of the economic rights over the Software.

Article 5 - SCOPE OF RIGHTS GRANTED

The Licensor hereby grants to the Licensee, who accepts, the following rights over the Software for any or all use, and for the term of the Agreement, on the basis of the terms and conditions set forth hereinafter.

Besides, if the Licensor owns or comes to own one or more patents protecting all or part of the functions of the Software or of its components, the Licensor undertakes not to enforce the rights granted by these patents against successive Licensees using, exploiting or modifying the Software. If these patents are transferred, the Licensor undertakes to have the transferees subscribe to the obligations set forth in this paragraph.

5.1 RIGHT OF USE – The Licensee is authorized to use the Software, without any limitation as to its fields of application, with it being hereinafter specified that this comprises:

1. permanent or temporary reproduction of all or part of the Software by any or all means and in any or all form.
2. loading, displaying, running, or storing the Software on any or all medium.
3. entitlement to observe, study or test its operation so as to determine the ideas and principles behind any or all constituent elements of said Software. This shall apply when the Licensee carries out any or all loading, displaying, running, transmission or storage operation as regards the Software, that it is entitled to carry out hereunder.

5.2 RIGHT OF MODIFICATION – The right of modification includes the right to translate, adapt, arrange, or make any or all modifications to the Software, and the right to reproduce the resulting software. It includes, in particular, the right to create a Derivative Software.

The Licensee is authorized to make any or all modification to the Software provided that it includes an explicit notice that it is the author of said modification and indicates the date of the creation thereof.

5.3 RIGHT OF DISTRIBUTION – In particular, the right of distribution includes the right to publish, transmit and communicate the Software to the general public on any or all medium, and by any or all means, and the right to market, either in consideration of a fee, or free of charge, one or more copies of the Software by any means.

The Licensee is further authorized to distribute copies of the modified or unmodified Software to third parties according to the terms and conditions set forth hereinafter.

5.3.1 DISTRIBUTION OF SOFTWARE WITHOUT MODIFICATION – The Licensee is authorized to distribute true copies of the Software in Source Code or Object Code form, provided that said distribution complies with all the provisions of the Agreement and is accompanied by:

1. a copy of the Agreement,
2. a notice relating to the limitation of both the Licensor's warranty and liability as set forth in Articles 8 and 9,

and that, in the event that only the Object Code of the Software is redistributed, the Licensee allows effective access to the full Source Code of the Software at a minimum during the entire period of its distribution of the Software, it being understood that the additional cost of acquiring the Source Code shall not exceed the cost of transferring the data.

5.3.2 DISTRIBUTION OF MODIFIED SOFTWARE – When the Licensee makes an Integrated Contribution to the Software, the terms and conditions for the distribution of the resulting Modified Software become subject to all the provisions of this Agreement.

The Licensee is authorized to distribute the Modified Software, in source code or object code form, provided that said distribution complies with all the provisions of the Agreement and is accompanied by:

1. a copy of the Agreement,
2. a notice relating to the limitation of both the Licensor's warranty and liability as set forth in Articles 8 and 9,

and that, in the event that only the object code of the Modified Software is redistributed, the Licensee allows effective access to the full source code of the Modified Software at a minimum during the entire period of its distribution of the Modified Software, it being understood that the additional cost of acquiring the source code shall not exceed the cost of transferring the data.

5.3.3 DISTRIBUTION OF DERIVATIVE SOFTWARE – When the Licensee creates Derivative Software, this Derivative Software may be distributed under a license agreement other than this Agreement, subject to compliance with the requirement to include a notice concerning the rights over the Software as defined in Article 6.4. In the event the creation of the Derivative Software required modification of the Source Code, the Licensee undertakes that:

1. the resulting Modified Software will be governed by this Agreement,
2. the Integrated Contributions in the resulting Modified Software will be clearly identified and documented,
3. the Licensee will allow effective access to the source code of the Modified Software, at a minimum during the entire period of distribution of the Derivative Software, such that such modifications may be carried over in a subsequent version of the Software; it being understood that the additional cost of purchasing the source code of the Modified Software shall not exceed the cost of transferring the data.

5.3.4 COMPATIBILITY WITH THE CeCILL LICENSE – When a Modified Software contains an Integrated Contribution subject to the CeCILL license agreement, or when a Derivative Software contains a Related Module subject to the CeCILL license agreement, the provisions set forth in the third item of Article 6.4 are optional.

Article 6 - INTELLECTUAL PROPERTY

6.1 OVER THE INITIAL SOFTWARE – The Holder owns the economic rights over the Initial Software. Any or all use of the Initial Software is subject to compliance with the terms and conditions under which the Holder has elected to distribute its work and no one shall be entitled to modify the terms and conditions for the distribution of said Initial Software.

The Holder undertakes that the Initial Software will remain ruled at least by this Agreement, for the duration set forth in Article 4.2.

6.2 OVER THE INTEGRATED CONTRIBUTIONS – The Licensee who develops an Integrated Contribution is the owner of the intellectual property rights over this Contribution as defined by applicable law.

6.3 OVER THE RELATED MODULES – The Licensee who develops a Related Module is the owner of the intellectual property rights over this Related Module as defined by applicable law and is free to

choose the type of agreement that shall govern its distribution under the conditions defined in Article 5.3.3.

6.4 NOTICE OF RIGHTS – The Licensee expressly undertakes:

1. not to remove, or modify, in any manner, the intellectual property notices attached to the Software;
2. to reproduce said notices, in an identical manner, in the copies of the Software modified or not;
3. to ensure that use of the Software, its intellectual property notices and the fact that it is governed by the Agreement is indicated in a text that is easily accessible, specifically from the interface of any Derivative Software.

The Licensee undertakes not to directly or indirectly infringe the intellectual property rights of the Holder and/or Contributors on the Software and to take, where applicable, vis-à-vis its staff, any and all measures required to ensure respect of said intellectual property rights of the Holder and/or Contributors.

Article 7 - RELATED SERVICES

7.1 – Under no circumstances shall the Agreement oblige the Licensor to provide technical assistance or maintenance services for the Software.

However, the Licensor is entitled to offer this type of services. The terms and conditions of such technical assistance, and/or such maintenance, shall be set forth in a separate instrument. Only the Licensor offering said maintenance and/or technical assistance services shall incur liability therefor.

7.2 – Similarly, any Licensor is entitled to offer to its licensees, under its sole responsibility, a warranty, that shall only be binding upon itself, for the redistribution of the Software and/or the Modified Software, under terms and conditions that it is free to decide. Said warranty, and the financial terms and conditions of its application, shall be subject of a separate instrument executed between the Licensor and the Licensee.

Article 8 - LIABILITY

8.1 – Subject to the provisions of Article 8.2, the Licensee shall be entitled to claim compensation for any direct loss it may have suffered from the Software as a result of a fault on the part of the relevant Licensor, subject to providing evidence thereof.

8.2 – The Licensor's liability is limited to the commitments made under this Agreement and shall not be incurred as a result of in particular: (i) loss due the Licensee's total or partial failure to fulfill its obligations, (ii) direct or consequential loss that is suffered by the Licensee due to the use or performance of the Software, and (iii) more generally, any consequential loss. In particular the Parties expressly agree that any or all pecuniary or business loss (i.e. loss of data, loss of profits, operating loss, loss of customers or orders, opportunity cost, any disturbance to business activities) or any or all legal proceedings instituted against the Licensee by a third party, shall constitute consequential loss and shall not provide entitlement to any or all compensation from the Licensor.

Article 9 - WARRANTY

9.1 – The Licensee acknowledges that the scientific and technical state-of-the-art when the Software was distributed did not enable all possible uses to be tested and verified, nor for the presence of possible defects to be detected. In this respect, the Licensee's attention has been drawn to the risks associated with loading, using, modifying and/or developing and reproducing the Software which are reserved for experienced users.

The Licensee shall be responsible for verifying, by any or all means, the suitability of the product for its requirements, its good working order, and for ensuring that it shall not cause damage to either persons or properties.

9.2 – The Licensor hereby represents, in good faith, that it is entitled to grant all the rights over the Software (including in particular the rights set forth in Article 5).

9.3 – The Licensee acknowledges that the Software is supplied "as is" by the Licensor without any other express or tacit warranty, other than that provided for in Article 9.2 and, in particular, without any warranty as to its commercial value, its secured, safe, innovative or relevant nature.

Specifically, the Licensor does not warrant that the Software is free from any error, that it will operate without interruption, that it will be compatible with the Licensee's own equipment and software configuration, nor that it will meet the Licensee's requirements.

9.4 – The Licensor does not either expressly or tacitly warrant that the Software does not infringe any third party intellectual property right relating to a patent, software or any other property right. Therefore, the Licensor disclaims any and all liability towards the Licensee arising out of any or all proceedings for infringement that may be instituted in respect of the use, modification and redistribution of the Software. Nevertheless, should such proceedings be instituted against the Licensee, the Licensor shall provide it with technical and legal assistance for its defense. Such technical and legal assistance shall be decided on a case-by-case basis between the relevant Licensor and the Licensee pursuant to a memorandum of understanding. The Licensor disclaims any and all liability as regards the Licensee's use of the name of the Software. No warranty is given as regards the existence of prior rights over the name of the Software or as regards the existence of a trademark.

Article 10 - TERMINATION

10.1 – In the event of a breach by the Licensee of its obligations hereunder, the Licensor may automatically terminate this Agreement thirty (30) days after notice has been sent to the Licensee and has remained ineffective.

10.2 – A Licensee whose Agreement is terminated shall no longer be authorized to use, modify or distribute the Software. However, any licenses that it may have granted prior to termination of the Agreement shall remain valid subject to their having been granted in compliance with the terms and conditions hereof.

Article 11 - MISCELLANEOUS

11.1 EXCUSABLE EVENTS – Neither Party shall be liable for any or all delay, or failure to perform the Agreement, that may be attributable to an event of force majeure, an act of God or an outside cause, such as defective functioning or interruptions of the electricity or telecommunications networks, network paralysis following a virus attack, intervention by government authorities, natural disasters, water damage, earthquakes, fire, explosions, strikes and labor unrest, war, etc.

11.2 – Any failure by either Party, on one or more occasions, to invoke one or more of the provisions hereof, shall under no circumstances be interpreted as being a waiver by the interested Party of its right to invoke said provision(s) subsequently.

11.3 – The Agreement cancels and replaces any or all previous agreements, whether written or oral, between the Parties and having the same purpose, and constitutes the entirety of the agreement between said Parties concerning said purpose. No supplement or modification to the terms and conditions hereof shall be effective as between the Parties unless it is made in writing and signed by their duly authorized representatives.

11.4 – In the event that one or more of the provisions hereof were to conflict with a current or future applicable act or legislative text, said act or legislative text shall prevail, and the Parties shall make

the necessary amendments so as to comply with said act or legislative text. All other provisions shall remain effective. Similarly, invalidity of a provision of the Agreement, for any reason whatsoever, shall not cause the Agreement as a whole to be invalid.

11.5 LANGUAGE – The Agreement is drafted in both French and English and both versions are deemed authentic.

Article 12 - NEW VERSIONS OF THE AGREEMENT

12.1 – Any person is authorized to duplicate and distribute copies of this Agreement.

12.2 – So as to ensure coherence, the wording of this Agreement is protected and may only be modified by the authors of the License, who reserve the right to periodically publish updates or new versions of the Agreement, each with a separate number. These subsequent versions may address new issues encountered by Free Software.

12.3 – Any Software distributed under a given version of the Agreement may only be subsequently distributed under the same version of the Agreement or a subsequent version.

Article 13 - GOVERNING LAW AND JURISDICTION

13.1 – The Agreement is governed by French law. The Parties agree to endeavor to seek an amicable solution to any disagreements or disputes that may arise during the performance of the Agreement.

13.2 – Failing an amicable solution within two (2) months as from their occurrence, and unless emergency proceedings are necessary, the disagreements or disputes shall be referred to the Paris Courts having jurisdiction, by the more diligent Party.

Version 1.0 dated 2006-09-05.

Bibliography

[1] Object-oriented methodology for software development with PELICANS. Reference Documentation of PELICANS.

Contents