

Drakkar Config Manager: How to manage and share various model configurations, based on NEMO-OPA9

J.M. Molines ^{*}, A.M. Treguier, S. Theetten [†]

Last update: Rev: 18 Date: 2007-07-02 12:49:40 +0200 (Mon, 02 Jul 2007)

1 Introduction

Within the Drakkar project, many model configurations (ORCA05, ORCA025, ORCA2, NATL3, ATL4, NATL4, NATL025, NATL12 ...) will be used and shared between various groups (LEGI, LPO, IFM-Geomar, Mercator, LOCEAN, Shirshov Institute of Oceanology-SIO). From previous experience in the Clipper project, we agree to have a common way of managing these configurations. The present document presents the concept of the Drakkar Config Manager (DCM), and can be used as a user manual, suitable for beginners.

2 General strategy

2.1 Sources management

The Drakkar project is based on NEMO-OPA9, and members of this project were chosen as beta-testers for this release. The official release of NEMO-OPA9 took place in April 2005, and some improvement/changes were introduced in DCM at this occasion. Release 1.12 of NEMO is the first release including AGRIF capabilities. It has been made available in Spring 2006. The introduction of AGRIF into NEMO, had a rather strong impact on the way the code is compiled etc... For DCM end users, the changes are made almost invisible. In this document, we assume that the reader is familiar with OPA9 and its structure as it is distributed by the ESOPA team. Documentation about the “modipsl” structure is found at the IPSL web site (see also Fig.??). The last release of NEMO (2.3beta) is the first release where input files are managed by a completely new module: IOM. This change of course brought compatibility issues with previous release, as the format for restart files has been modified. With respect to DCM, basically, it ends up with the management of a new sub-directory (IOM). Documentation about NEMO-OPA9 is available on the NEMO web page (LOCEAN).

The strategy for the sources management, for a given configuration, is based on the idea of having only OPA9 modules differing from “a” reference version, hoping that they are not too numerous. These modules are stored under a private area, under the responsibility of the particular configuration “owner”. Within the project we distinguish between the CVS reference version (the one distributed and updated under CVS by ESOPA team) and the DRAKKAR reference version which includes:

- The modules which corresponds to Drakkar configurations and that will probably never be in the CVS reference version (thinking for example of the *par_ATL4.h90* module)

^{*}Laboratoire des Ecoulements Géophysiques et Industriels, CNRS UMR 5519, Grenoble, France

[†]Laboratoire de Physique des océans, CNRS-Ifremer-UBO, Brest, France

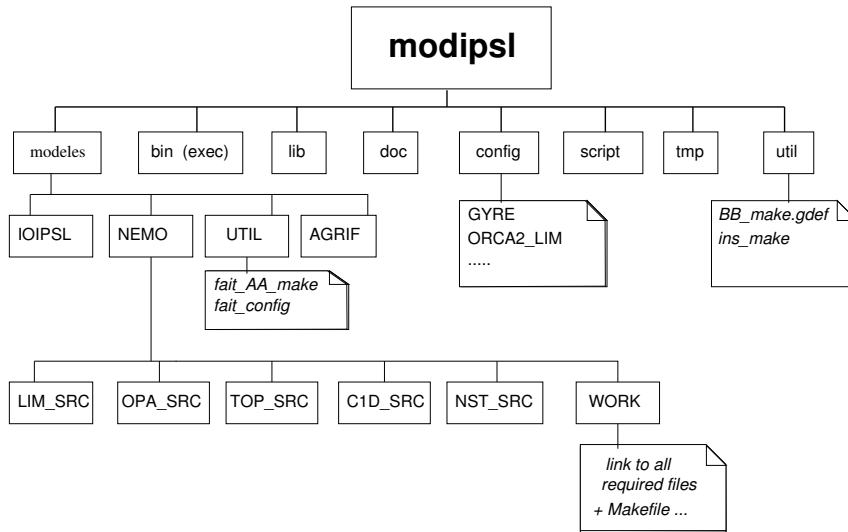


Figure 1: Structure of the modipl tree.

- The modules with recent bug fixes not already available on the CVS reference version.
- Also the particular tools such as *fait_AA_make* or *ins_make* if they differ from the official ones.

The rationales to have as few as possible “private” modules are manifold:

- limit the amount of disk space on the host computer
- easier code maintenance as bugs can be corrected for many configurations at once
- increase of the readability of the configuration: just looking at the modified modules, one knows what is special in this particular configuration or sub-configuration

In order to compile the code, for a given configuration, it is necessary to have all the required modules in a `WORK` directory, with the correct Makefile. This is done from the private configuration directory by a special “makefile” (described in detail below), which mainly builds the `WORK` directory by copying the sources from the different levels in the following order:

- the CVS reference version
- the DRAKKAR reference version
- the configuration reference version
- the current working version.

The `WORK` directory of the current working version is located within a full “modipsl” directory tree, because only in that way can the standard modipsl tools for making the Makefile be used. This tree can be considered as “volatile” as it can be easily reconstructed from the reference directories. It is typically located on a large file system. (At IDRIS, for instance, it can be either on the `WORKDIR` or on the `HOME_BIS` directory, depending on the host computer).

To make things easier for the user, it is advisable to define variables for the NEMO reference (variable `REFDIR`) and the DRAKKAR customized version (variable `CUSDIR`). Several script examples (`dmgrrc_zahir.csh`, `dmgrrc_zahir.ksh`, `dmgrrc_rhodes.csh` or `dmgrrc_rhodes.ksh`) are provided to set the environment variables.

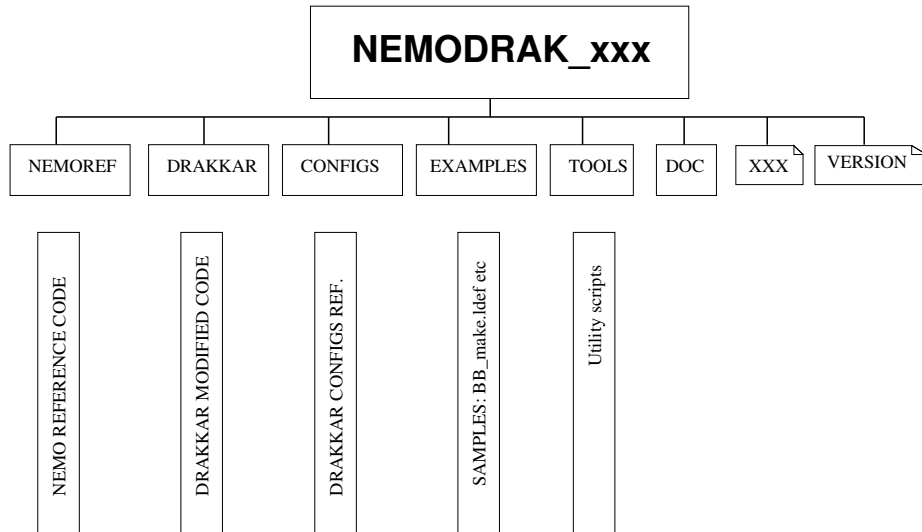


Figure 2: Structure of the modispl tree.

From a practical point of view, DCM is a directory, usually called `NEMODRAK_1.12` or `NEMODRAK_2.3dev` for instance, where the number after `NEMODRAK` is related to the NEMO reference version. Figure ??, indicates the basic structure of a `NEMODRAK` directory.

2.2 Run management

2.2.1 Machines hierarchy

Run management is somewhat easier than sources code management. It is mainly a question of how to name the files and directories relative to a particular run. As a starting point we have to consider 3 kinds of machines:

1. the compilation machine where the executable code is build (hereafter the **C** machine). This machine differs from the production machine when a cross-compiler is used, as it is the case with the SX NEC computer. On this machine we will assume that some environment variables are set: `UDIR` (user directory, usually `$HOME`), `CDIR` (compile directory, where the “volatile” `WORK` directory and its copy of `modispl` are kept), as well as `PDIR` and `SDIR` which respectively points toward the `HOME` of the production machine and the `HOME` of the storage machine, eventually via NFS.
2. the production machine where the executable code is run (hereafter the **P** machine). Production and compilation machine can be the same (case of the IBM p690, for instance). On this machine we will assume that environment variables `HOME` and `USER` are set.
3. the mass storage machine where all the files (forcing, results) are archived (hereafter the **S** machine). In a general case, this machine is different from the previous ones as `OGCM` produce a large amount of data. But conceptually, this can be virtually any machine. On this machine we will assume that environment variables `HOME` and `USER` are set.

2.2.2 Naming rules

The configuration names are defined in two parts: the `CONFIG` name by itself, followed by the `CASE` name, separated by a `'.'`. Example : `ORCA05-K001` or `ATL4-B04`. Note that the case

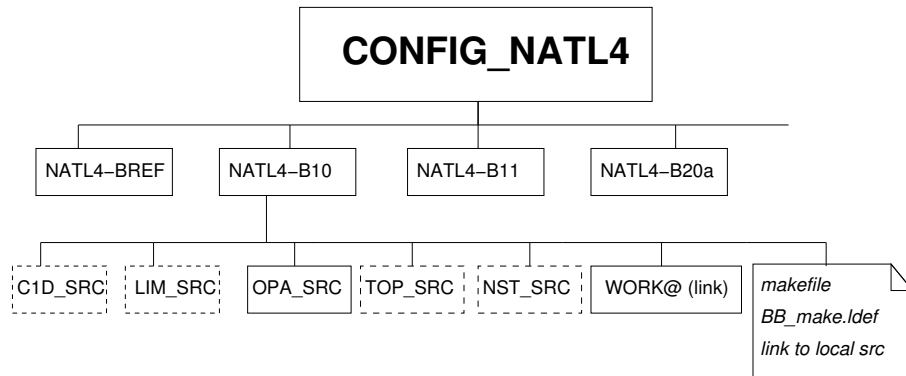


Figure 3: Structure of the configuration tree on the user directory of the C machine: example for NATL4 config. LIM_SRC (ice model), TOP_SRC (tracer model), C1D_SRC (1D model) or NST_SRC (nesting) are indicated in dash lines as they are optional; however, they are still automatically created, even if they are empty.

name must start with a letter identifying the group which is responsible for this case. This is a recommendation within the DRAKKAR group but virtually any name can be used, of course!

- B stands for Brest (LPO group)
- H stands for Helsinki (University of Helsinki group)
- K stands for Kiel (IfM-geomar group)
- G stands for Grenoble (LEGI group)
- M stands for Moscow (SIO group)
- P stands for Paris (LODYC group)
- T stands for Toulouse (Mercator group)

As a generic name we will refer hereafter to *CONF-CASE*.

2.2.3 Directory tree

- On **C** machine :
 - + \$UDIR/CONFIG_CONF where all the cases will be kept (Fig. ??).
 - + \$UDIR/CONFIG_CONF/CONF-CASE/ where the private files will be kept. We will refer to this directory as the **current tree**.
 - + \$CDIR/WCONF-CASE where a modipsl tree will be build. We will refer to this directory as the **working tree**.
- On **P** machine :
 - + \$PDIR/RUN_CONF/CONF-CASE

- + \$PDIR/RUN_CONF/CONF-CASE/EXE where the executable will be copied from **C**
 - + \$PDIR/RUN_CONF/CONF-CASE/CTL where the job scripts are stored and will be launched. A template for the namelist file will also be here and eventually other control file for the run.
- On **S** machine :
- + \$SSDIR/CONF/CONF-I : Initial conditions, bathymetry, grid etc...
 - + \$SSDIR/CONF/CONF-CASE-S : Storage of results
 - + \$SSDIR/CONF/CONF-CASE-R : Restart files
 - + \$SSDIR/CONF/CONF-CASE-MEAN : Other directories are/will be constructed in parallel with these, to hold results from diagnostic programs. For example MEAN subdirectory will hold the temporal mean fields for a run (monthly, quarterly, annual...).

3 Implementation

The implementation of the sources management strategy has two levels: The “administrator” level and the “user” level.

3.1 Administrator level

- Maintenance of the CVS reference \$REFDIR (performing updates when available)
- Maintenance of the DRAKKAR reference \$CUSDIR, either adding a corrected module when a bug fix is reported or deleting module when the fix has been included in the CVS update.
- Install the set of user scripts (under NEMODRAK/TOOLS) and the templates for makefile and BB_make.1def (under NEMODRAK/EXAMPLES) The user scripts differ from one machine to the other, depending on the configuration of NFS between the machine hierarchy. But the idea is to maintain only one user interface, so that the user tasks are the same on any platform.

On each platform then, CVS references and DRAKKAR references must be accessible by the members of the group in read access. Since January 2007, we decided to maintain DCM under version control, using subversion. It greatly improves the maintenance of DCM. The current version of DCM can be 'checked out' at :

```
svn://meolcar.hmg.inpg.fr/home/forge/REPOS_DCM/DCM/trunk
```

3.2 User level

At the user level, implementing a new configuration can be done in few steps, with the help of available user scripts and makefile, always working on the **C** machine:

1. **Set some environment variables, in the starting files.**

Depending on the working shell, the environment must be customized, (in `.cshrc` or `.profile`) in order to fix the proper values to some environment variables, which indicates the root of the directory system to be used.

- HOMEDCM:** This is the name of the root directory for the Drakkar Config Manager. Usually for one platform every user points to the same HOMEDCM, (for instance, /home-gpfs/rech/cli/rci002/NEMODRAK_1.12). Note that since NEMO tag 1.06, the base-name of HOMEDCM is NEMODRAK_tag. Doing this way, it is much easier to know which official version DCM is based on. It also allows the re-building of older versions.
- UDIR:** **U**ser **DIR**rectory. This is usually \$HOME but can be any convenient and safe directory, where the user developpement will be stored.
- CDIR:** **C**ompile **DIR**rectory. This is where the whole modipls tree will be build in order to compile a particular configuration. It needs a relatively large amount of space. It is usually \$HOME_BIS or \$WORKDIR on IDRIS machines. Files under this directory can be re-build easily.
- PDIR:** **P**roduction **DIR**rectory. This the directory from where the executable will be launched, via job submission stuff.
- SDIR:** **S**torage **DIR**rectory. This is usually the \$HOME directory on the **S** machine, with a very large file system to archive the results of a simulation. At IDRIS it is, for example \$HOMEGAYA (accessible from rhodes via NFS, or via rcp from zahir)
- REFDIR:** **REF**erence **DIR**rectory. The directory used to check-out the ESOPA CVS NEMO. It must be a strict copy of the official CVS release.
- CUSDIR:** **CUS**tom **DIR**rectory. The directory where the DRAKKAR modified files are stored, in the same directory tree than the official release. Files under this directory are either DRAKKAR dependent, (valid for all DRAKKAR configurations) or files with bug fixes not already reported in the official release.

This setting is to be done once, at the first installation of the system.

2. **Update the \$PATH variables to include \$HOMEDCM/TOOLS/ in it.** For instance, in .cshrc add a line such as `set path=($path $HOMEDCM/TOOLS/)`.
3. **Choose the correct alias to mkconfdir:** This script is used in the next step, but there are actually 3 variants of the script, depending on the platform. If all C, P and S machines can be seen as 1 single machine (via NFS, for instance) then


```
alias mkconfdir $HOMEDCM/TOOLS/mkconfdir_rhodes.
```

 If S is an independent machine, accessible through rsh only:


```
alias mkconfdir $HOMEDCM/TOOLS/mkconfdir_zahir.
```

 If you work at DKRZ:


```
alias mkconfdir $HOMEDCM/TOOLS/mkconfdir_cross.
```

 If you work at IFREMER:


```
alias mkconfdir $HOMEDCM/TOOLS/mkconfdir_nymphea.
```

4. **Create the directory tree on the system: mkconfdir**

This script takes two arguments : `mkconfdir CONF CASE`

It creates the directory tree as defined in 2.2.3 above. It uses the environment variables set in the previous point. If both the **P** machine and the **S** machine can be seen from **C** machine by NFS, there are no problems. If it is not the case, the script uses rsh command to perform directory creation on remote machine, and this action requires proper setting of the .rhosts file in the HOME directory of the remote hosts.

Additionally, this script copy templates files for BB_make.ldef and makefile to the CONFIG

directory (if they are not already in this directory). It sets the correct CONFIG and CASE lines in these files.

5. Check the `BB_make.ldef` and `makefile`

These files are copied in the previous step. It may be useful to start from a previous config. If so, just copy these files from the other config and edit them to fix options for the *CONF-CASE* configuration.

BB_make.ldef: Edit the lines corresponding to CONFIG and CASE.

Edit the lines corresponding to the compiler options (F_O).

Edit the lines corresponding to the CPP keys for your configuration.

Please note that the lines that will be taken into account are those quoted for the target system. For instance, lines taken for SX NEC are quoted `#-Q- sxnec`

makefile: This file is at the center of the configuration manager, and some lines must be carefully edited.

CONF: set this variable to the *CONF* name

CASE: set this variable to the *CASE* name

CASEREF: set this variable to reference CASE for this config. If no reference case are used, just set CASEREF to 'none'. Each time "gmake install" is done, the files from the reference case will be copied into WORK after the reference \$REFDIR and custom \$CUSDIR files, but before your present configuration's files. This capability is meant to help the user maintain variants of his basic configuration without having many copies of each modified source file.

PREV_CONFIG: set this variable to the *full path* of a previous configuration used as template, if you want to start a new configuration by copying all the files of a previous one. If no previous configuration is used, just set PREV_CONFIG to 'none'. This variable is used only with the `gmake copyconfig` target (see below). Example:

```
PREV_CONFIG = /home/rech/cli/rcli099/CONFIG_ATL4/ATL4-B03
```

MACHINE: set this variable to the type of system you are using: `sxnec` for SX NEC machine, `aix` for IBM, etc ...

OPA: must be set to 'use' (as far as you need OPA ...)

LIM: must be set to 'use' for NEMO_xxx. Even if you do not use it at run time.

C1D: must be set to 'use' for NEMO_xxx. Even if you do not use it at run time.

TOP: It is set to 'use' if you need the Tracer Model. To anything else if you don't.

AGRIF: It is set to 'use' if you need AGRIF. To anything else if you don't.

6. Create the working directories tree: `gmake install`

This will do a lot of things:

- Copy the reference files (REFDIR, CUSDIR, CASEREF) to the working tree
- make the adequate links to the WORK directory of the working tree.
- create a link to WORK in your *CONF-CASE* directory.
- launch the script that build the WORK Makefile, and install the Makefile.

(There are many other targets in the makefile, see below for a full description).

7. Customize your own code files : `getfile`

If you want to customize for your own configuration, you must put the files to customize in the current tree. This is easily done with the command script `getfile`

Usage: `getfile xxxx.F90`

This command assume that you are in the `WORK` directory. It will copy `xxx.F90` in the right position of your *CONF-CASE* tree, and make a link in the *CONF-CASE* in order to have a global view of all the customized file for your CASE.

For example, if you are working in the `ORCA025-G02`, and you want to get `ocesbc.F90` for customization, you just have to do

`cd WORK` : you use the link to the working tree.

`getfile ocesbc.F90` :

copy `WORCA025/modeles/OPA/OPA_SRC/SBC/ocesbc.F90` to `CONFIG_ORCA025/ORCA025_G02/OPA_SRC/SBC/ocesbc.F90` and make a link to this file in `CONFIG_ORCA025/ORCA025_G02/`

8. Compile your code, type `gmake`

Each time a change is done in your current tree you have to repeat this last step, only.

The following appendices describes some technical details that can be of some help for the user who wants to set up model configurations and use the system deccribed in this report.

A Setting the environment

In order to work properly, many tools rely on the proper setting of some environment variables that must be done previous any other things. In this first appendix, we describe the setting for `cs`h users or `ks`h users. All the tools described in this section are to be used on the **C**ompilation machine. They are available in the `TOOLS` directory, besides the Reference tree.

A.1 *dmgrrc_machine.csh*

This is a short script file for setting the environment for `cs`h users, as said in 3.2.1 above. It must be edited to fit the actual user situation. It has to be included in the `.cshrc`. The use of aliases may facilitate the navigation between the different directory tree.

It is also recommended to make aliases to navigate into the directory trees. Examples:

```
# Optional (but recommended aliases) Edit as you like
# CONFIG_ORCA025
#=====
# G01
# alias vg1 'cd $HOME/CONFIG_ORCA025/ORCA025-G01'
# alias wg1 'cd $HOME_BIS/WORCA025-G01/modeles/OPA/WORK'
# alias cg1 'cd $PDIR/ORCA025-G01/CTL'
#
# G02
# alias vg2 'cd $HOME/CONFIG_ORCA025/ORCA025-G02'
# alias wg2 'cd $HOME_BIS/WORCA025-G02/modeles/OPA/WORK'
# alias cg2 'cd $PDIR/ORCA025-G02/CTL'
#
# ...
```

A.2 *dmgrrc_machine.ksh*

This is a short script file for setting the environment for `ks`h users, as said in 3.2.1 above. It must be edited to fit the actual user situation. It can be included in the `.profile`.

It is also recommended to make aliases to navigate into the directory trees. Examples:

```
# Optional (but recommended aliases) Edit as you like
# CONFIG_ORCA025
#=====
# G01
# alias vg1='cd $HOME/CONFIG_ORCA025/ORCA025-G01'
# alias wg1='cd $HOME_BIS/WORCA025-G01/modeles/OPA/WORK'
# alias cg1='cd $PDIR/ORCA025-G01/CTL'
#
# G02
# alias vg2='cd $HOME/CONFIG_ORCA025/ORCA025-G02'
# alias wg2='cd $HOME_BIS/WORCA025-G02/modeles/OPA/WORK'
# alias cg2='cd $PDIR/ORCA025-G02/CTL'
```

```
#  
# ...
```

B Description of the available tools and scripts

B.1 *mkconfdir*

Usage : `mkconfdir CONF CASE`

CONF = configuration name (*eg* ORCA2, NATL4, ORCA025 ...)

CASE = case id (*eg* G01, KAB002, B02 ...), respecting the rules edicted in previous section 2.2.2

Create the directory tree for a given CONF CASE. It prepares the directories on the **C**ompiler machine, the **P**roduction machine and the **S**torage machine. Directory creation is either done through NFS, or via rsh commands from the **C** machine. In fact there are 2 variants of `mkconfdir` and the `mkconfdir` must be linked to the appropriate one:

- **P** and **S** machines are seen via NFS from **C** or
 C and **P** and **S** is the same machine: `mkconfdir_rhodes`
- **C** and **P** is the same machine, **S** is accessible via rsh: `mkconfdir_zahir`
 In this case, the `.rhosts` file for the **S** machine must allow rsh connection from **C**.

Other cases can be handled easily from these standard cases. Additionally `mkconfdir` copies the template `BB.make.ldef` and `makefile` from the `EXAMPLES` directory (if they don't already exist), and set the correct `CONFIG` and `CASE` variables in these 2 files.

B.2 *makefile*

This `makefile` can be used for many purposes:

B.2.1 `gmake`:

Action: compile the code for the current tree.

The current way of compiling the code.

B.2.2 `gmake install`:

Action: Create and initialize the working tree.

Normally done only once, at the installation of a new CONF-CASE

B.2.3 `gmake reinstall`:

Action: Copy the source files to `WORK`, but do not try to make the `Makefile`

. Can be used if for any reason some of the files in `WORK` have been deleted. As `Makefile` is not reconstructed, it should still exists in `WORK`.

B.2.4 `gmake reinstall`:

Action: This is almost like `gmake install`, but does't build the `Makefile`. (Much faster).

This action is usefull when the user wants to update the source code after an upgrade of the `Drakkar Manager`. As the `Makefile` is not re-build, it is assumed that (1) it exists, (2) dependencies are not changed from the existing `Makefile`. If after a `gmake reinstall` the code fails to compile, it is better and safe to do a `make install` again.

B.2.5 `gmake copyconfig`:

Action: copy the configuration pointed as `PREV_CONFIG` to the current `CONF-CASE`.

B.2.6 `gmake restore`:

Usage: `make restore RESTORE=OPA_SRC/SOL/solsor.F90`, for example.

The full path for the module to be restored must be given relative to the `OPA` directory.

This target is used automatically in the `rmfile` script (see below).

B.2.7 `gmake clean`:

Action: execute a `gmake clean` in the `WORK` directory.

All modules and library will be erased. All sources will be re-compiled.

B.2.8 `gmake cleaninst`:

Action: Delete the working tree from `$CDIR`.

Use with caution. Need a `gmake install` to restore the working tree.

B.2.9 `gmake links`:

Action: rebuild the links into the `CONFIG_CASE/CONFIG-CASE/` directory.

This is normally done when compiling, but it can be useful if some source code is added manually.

B.3 *getfile*

Usage: `getfile xxxxx.?90 [-d | -r]`

`xxxx.?90` : any fortran file in the `WORK` directory.

Make a copy of this file in the current tree. Add a link at the root of the current tree. If not in a `WORK` directory, try to `'cd WORK'` and exit if it cannot. `CONF` and `CASE` names, are deduced from the full path of the `WORK` directory. The location of the module in the `OPA` tree is deduced from the link.

(*eg* : `limadv.F90 ->../LIM_SRC/limadv.F90` or
`trasbc.F90 ->../OPA_SRC/TRA/trasbc.F90`)

An extension to this function is available with either the `'-d'` or the `'-r'` switch. With `'-d'` `getfile` retrieves the file directly from the `DRAKKAR` directory, with `'-r'`, from the `NEMOREF` directory. This is the best way to upload a reference module, bypassing the different layers.

B.4 *rmfile*

Usage: `rmfile xxxxx.?90`

Remove a file from the current tree. This command must be issued from the root of the current tree. After asking for confirmation (twice), it restores the removed file from the `REFERENCES` (`REFDIR/CUSDIR/CASEREF`), it deletes the local link to `xxx.?90` and move the file from its actual position in the tree to the `.trash` directory (this directory is created if it does'nt exist). It is the user responsibility to empty the `.trash` directory! The restoring of the reference version is made by `make restore RESTORE=xxxx.?90`.

B.5 *putfile*

Usage: `putfile xxxxx.?90`

This function needs administrator privilege to be used: It copies the XXXX.?90 fortran file from a local CONFIG-CASE to the DRAKKAR directory corresponding to HOMEDCM. This is very usefull when phasing a code, or for maintenance of DCM

B.6 *cmpfile*

Usage: `cmpfile xxxx.?90 [-t]`

Easy way for making either a 'xdiff' or a 'diff' between a file in the current tree and the corresponding file in the reference tree. Default option launches xdiff whether `-t` option produces a `diff -bic` listing. It can be used as template for personal improvement. If the target file is also into the DRAKKAR DCM tree, then a warning is issued. And the user may find usefull to use `dcmpfile` too (see below).

B.7 *dcmpfile*

Usage: `dcmpfile xxxx.?90 [-t]`

This is almost the same than `cmpfile`, except that the difference is done with respect to the DRAKKAR tree, not the NEMOREF tree. Very usefull too.

B.8 *Dcmpfile*

Usage: `Dcmpfile xxxx.?90 [-t]`

Again a variant of the previous `cmpfile`... This is usefull for maintenance, as it gives the difference between DRAKKAR and NEMOREF. It must be run within the DRAKKAR tree. (This can be done by any user, safely. No write access are granted to 'standard' user).

B.9 *mkdifdir*

Usage: `mkdifdir [-short]` (Must be used within a WORK directory)

This script creates a file (named `diflist.'date'`) with all the differences between the current working tree and the reference tree. The difference is made with the command `diff -bic` and the resulting file is quite easy to look at with `vi`, or `gvim` with color enhancement that outlines differences.

When option `-short` is specified, the diff is limited to the OPA source code, excluding the IOIPSL, UTIL and util directories. This command is still in the distribution but not checked for a while ...

B.10 *lookforkey90*

Usage: `lookforkey90 key_xxxxx`

Give a list of the modules (*.?90) of the current directory where the `key_xxx` is used, and indicates the number of occurences per modules. Example:

```
% lookforkey90 key_dtasal
```

```
Search modules with use of key: key_dtasal
```

```
Module Name : number of lines where key_dtasal appears
```

```
dtasal.F90 : 2 lines
```

```
istate.F90 : 1 lines
```

```
ocesbc.F90 : 2 lines
tradmp.F90 : 1 lines
```

C Templates files

This system depends on some files that must be customized by the user. In order to help this process we provide a set of template files that can be used as a starting point for user customization. These files are located in the directory **EXAMPLES** besides the reference tree.

- BB_make.ldef
- makefile
- namelist
- namelistio

D Reference Configurations

Starting with version 1.06 of DCM, reference DRAKKAR versions (*e.g.* NATL4, ORCA025), in phase with the actual NEMO version can be found in the CONFIGS directory. In this directory, there is also an example of the corresponding namelists. These reference configs, can be used as CASEREF in the makefile. To do so, you need to make a link to these directories under your CONFIG_CONF directory. For instance :

```
rcli002@zahir001 >20 pwd
/homegpfs/rech/cli/rcli002/CONFIG_ORCA025
rcli002@zahir001 >21 ls -l ORCA025-GREF1.06
lrwxrwxrwx  1 rcli002  cli                66 Oct 19 11:39
ORCA025-GREF1.12 -> /homegpfs/rech/cli/rcli002/NEMODRAK_1.12/CONFIGS/ORCA025-GREF1.12/
```

In this latter example, in the makefile for config ORCA025-G70, CASEREF is defined to GREF1.12.