ICHEC Harmonie User Guide

1.0 Introduction

This guide is intended to help anyone approaching the Harmonie model for the first time. For the benefit of the uninitiated, it attempts to explain the mechanics of running Harmonie as well as the organization and design of the system. I don't write from any deep understanding of Harmonie, just as someone who has learned a little about it through trial and error, and in the hope that this may help others avoid some of the traps that I fell into.

This guide complements and, to some extent, duplicates other on-line Harmonie documentation, e.g.,

- http://hirlam.org/index.php (see "General description of the HARMONIE model" under "POPULAR" listing on right side of page).
- https://hirlam.org/trac/wiki/Harmonie_36h1
- https://hirlam.org/trac/wiki/HarmonieSystemDocumentation (and all the links included here).

Once registered as a bona fide Hirlam/Harmonie user, those guides may be all you need. If not, however, you may want to read on.

2.0 Harmonie is a "system"

Harmonie is not just a collection of source-files that you compile and link into an executable. It is a complete system of source-files, scripts, libraries, executables, and data repositories, all designed to routinely (and largely automatically) process observations, assimilate them into "analyses" of the state of the atmospheric system over the Europe and the North Atlantic, run the forecasting model, and post-process the output. The system is designed to run largely without interacting with humans, and the design makes sense when viewed in that way. As a corollary, however, it can be difficult for a human to control the system through the deep hierarchy of Harmonie scripts. Nevertheless, it can be done, and this document is intended to help in so doing.

2.1 Harmonie System Components

Like any weather forecasting model system, Harmonie consists of the following broad components:

1. ***Source-code files for the weather model itself***, expressing the dynamics and physics of the atmosphere in numerical algorithms, primarily written in Fortran90.
2. ***Source code for various pre- and post-processing executables***, such as those needed to convert data between generic compressed

formats and formats specific to each particular model domain grid and resolution.

3. ***Input data*** of various kinds and in various formats, including:
    a. "*Climate*" or long-term data (such as sea-surface temperature or SST, topographic heights, land surface properties, etc.),
    b. "*Boundary*" data, typically output from a larger regional or global model (the "nesting" model), which is used to initialize and constrain the boundaries of the Harmonie model,
    c. *Observational* data, also assimilated into Harmonie to provide initial conditions for the start of each run.
    d. *Output data from previous Harmonie runs.* Optionally, this can be used to supplement boundary data from the larger nesting model.
4. ***Output data***, also of various kinds:
    a. "Raw" output fields generated by the forecast model as it runs;
    b. "Post-processed" data, possibly in different format and on a different grid to the raw data.
5. ***Control files***, i.e., a large set of shell and perl scripts that control and co-ordinate the model build, and also the pre-processing, data assimilation, forecasting, and post-processing phases of each model run. In the case of Harmonie, "control" is very much the operative word. The scripts can be quite dense and multi-layered, and they can subtly resist human intervention to change their behavior! You must decide whether to try to beat them or to join them. The better option is probably to try to understand them as much as possible, and work with them.

2.2 Harmonie directories

In a working Harmonie system, files are distributed over several separate directory hierarchies. The main directories are listed below. Most initial work is done in no. 1 (HARMONIE_ROOT). There is no need to worry about 2-4 below until after the code is built:

1. **HARMONIE_ROOT**: this is the top-level directory containing the reference Harmonie source-code installation, or repository, and the libraries and executables built from those sources. It is specified in config-sh/config.<your-compcentre-suffix> (described in section 3.3 below). Think of this as a relatively fixed set of files that don't change very much or very often. This is your starting point.
2. **HM_HOME** or **home**: home of your personal experiment runs, most likely somewhere under your $HOME directory, in a "fast permanent file system" (FPFS, in Harmonie jargon). Each experiment is run from $HM_HOME/$EXP. Files in here tend to be small configuration files for each run.
3. **hm_data** and **HM_DATA**: hm_data is the actual name of a directory you must explicitly create yourself, e.g., in a scratch or work filesystem (LVFS, or large volatile filesystem, in Harmonie jargon), which is normally also the fastest available. HM_DATA is then $SCRATCH/hm_data/$EXP. It is created automatically (if it doesn't

already exist). In a sense, hm_data is the “work-space” analogue to HM_HOME, while HM_DATA is the work-space analogue to HM_HOME/$EXP. Most of the real Harmonie action happens under $HM_DATA. $HARMONIE_ROOT is essentially copied to $HM_DATA and all the text files needed or generated at run-time are placed there.

4. **HM_ARC:** typically $SCRATCH/hm_arc, which you must explicitly create yourself, and which must exist before starting a Harmonie run. Moreover, you must also explicitly create $HM_ARC/$EXP for each experiment $EXP. These directories contain archive output from each experiment run. There is rarely any good reason for a human to look in there.

3.0 Harmonie Source Code Download

Go to an empty directory, and once registered as a Harmonie user, issue the Subversion “svn” command to get the particular Harmonie version desired (currently harmonie-36h1.3):

svn export https://svn.hirlam.org/tags/harmonie-36h1.3

Or, for the Met Eireann branch:

svn export https://svn.hirlam.org/branches/METIE/harmonie-36h1.3

This copies the basic Harmonie kit from the SVN repository into your own directory. This directory is referred to as HARMONIE_HOME in the control scripts.

For more information on Subversion, see “man svn”, or check out http://svnbook.red-bean.com/en/1.5/svn-book.pdf . The basic command is “svn” . The “svn” command accesses files in repositories, which are always accessed with URLs.

3.1 First Key Files to Edit

There are 4 files to check and possibly edit before starting to build anything. If $HARMONIE_HOME is the path to the top level Harmonie directory (containing config-sh, const, msms, nam, scr, src, sms, util), then the first key files to look at are:

```
$HARMONIE_HOME/util/makeup/config.<your-compcentre-suffix>
$HARMONIE_HOME/config-sh/config.<your-compcentre-suffix>
$HARMONIE_HOME/config-sh/submit.<your-compcentre-suffix>
$HARMONIE_HOME/sms/config_exp.h
```

In `$HARMONIE_HOME/config.sh` and `$HARMONIE_HOME/util/makeup`, there are large collections of `config.<your-compcentre-suffix>` files (and `submit.<your-compcentre-suffix>` files in `$HARMONIE_HOME/config.sh`). If you

start with the one (or one of those) corresponding to your own computer centre ($COMPCENTRE), all you may need to change are the paths to your own home and work-space directories. There are helpful comments in each `config.<your-compcentre-suffix>` file. However, here is some more information about some of the key macros or shell-variables which could be useful:

3.2 The "util/makeup/config.<your-compcentre-suffix>" File

This is the first user-configurable control-file that is invoked. It is used to control how the Harmonie libraries and executables are compiled and linked. It is very like a standard "make.inc" file, containing the usual macros for compilers and compiler options.

I found I did not have to change anything in the version of this file appropriate for my centre. Your experience may differ. If your Harmonie build fails, check the usual suspects in here (DEFS, COPT, FOPT, etc.) and make sure that the settings are compatible with your installed compilers and MPI.

It is possible to change the compiler and linker options provided here for others you may prefer (e.g., add "-pg" or just "-p" to instrument the code to generate profiling data, which is output in a gmon.out file that can be processed by gprof).

Just beware that some options suitable applications running on the back-end compute nodes (e.g., -xSSE4.2) may not work at all for applications running on the head node! It may be tempting to raise the compiler optimizations used in the supplied config file (e.g., from –O2 to –O3), but most likely this will turn out to be a bridge too far; the runs will fail somewhere for no obvious reason, and you will just have to revert to the original compiler options to get things working again.

3.3 The "config-sh/config.<your-compcentre-suffix>" File

This file specifies many user- and system-specific details, including paths to where most of the input data for Harmonie are to be found on your system. Many of the macros defined in this file are self-explanatory. Note the discinctions between:

- "HARMONIE" directories (i.e., local repository of the reference-set of source files, scripts, and executables);
- "WORK" or "SCRATCH" directories, used during run-time, containing volatile and possibly very short-lived files and data;
- "HOME" or "Experiment" directories, containing just the small control-files for each Harmonie run.

Otherwise, some key macro settings in this config.<your-compcentre-suffix>

(or “Env_system”) file include;

```
HM_ARC=<dir-path>          # Path to hm_arc directory, which should already
                           # exist. Will end up containing $EXP sub-
                           # directories, which in turn contain input climate
                           # data for each “experiment”.

HARMONIE_CONFIG=<suffix>   # where <suffix> is the suffix to the config.*
                           # files for your $COMPCENTRE.  All these config
                           # files should be consistently named!

HM_CLDATA=<path>           # where <path> is the path to the directory
                           # containing generic, raw, climate data to be used
                           # as “input” to be pre-processed for each different
                           # Harmonie experiment or configuration.

EXP==${EXP?"Give 'experiment' identifier EXP before running this script"}
                           # Leave this as is.  EXP is normally provided on
                           # the command-line (or “guessed” in a sensible
                           # way by Harmonie scripts)

HM_DATA=$lvfshost0/hm_data/$EXP    # This will be the top of the main
                           # working directory structure.

BUFR_TABLES=$HM_LIB/util/auxlibs/bufr_000381/bufrtables/
BUFRTAB_DIR=$HM_LIB/util/auxlibs/bufr_000381/bufrtables/
                           # These macros differ between the “metie-36h1.3” and
                           # the standard “harmonie-36h1.3” versions.

FULLSMS=no                 # Use “no” here, because we use mini-SMS instead.

PERL_TK=perl               # Should be enough to find all necessary perl stuff.
PERLTKDIR=                 # Leaving this empty works for me…

MAKEUP=<yes|no>#  “yes” to use the more modern “Makeup” build system;
                           # “no” to use the original “gmkfile” system.
                           #  Most users should say “yes” here.  This is the very
                           #  last line in the file, but just as important as any!
```

When you are setting up to run an experiment, this file is copied to the $HM_HOME/$EXP/config-sh directory, and soft-linked to Env_system in the $EXP directory immediately above.

3.4 The “config-sh/submit.<your-compcentre-suffix>” File

This file provides details of how to launch various kind of jobs (e.g., “scalar”, “background”, or “parallel”) on your particular system. It is used to generate

job-submission scripts for your particular batch system (e.g., PBS). The default version of this file uses particular batch system “accounts” and “queues”, so it is important to either get permissions to use those accounts and queues, or to change them to ones of your own.

Some key parameters set in this file include:

- $submit # Make sure this command is right for your system.
- $nnodes, $ntasks, $nprocx, $nprocy # nodes, tasks/node, and domain decomposition, for full parallel run of Harmonie forecast.
- Search for “ppn” in this script, and ensure it is appropriate for each system (e.g., ppn=8 for 8-core nodes).
- ‘ACCOUNT’ # Either get permissions for the account specified, or change the (PBS) account for one you can use.
- ‘QUEUE’ # Beware of pitfalls here, regarding queue permissions, and also min. and max. no. of nodes that each queue permits. E.g., the “ShortQ” on stokes only permits jobs with a 2-node minimum, so the “scalar” Harmonie scripts submitted to this queue will provoke an “sms abort”.
- ‘ZHOOKDIR’, or DR_HOOK_PROFILE; set this to an empty directory where profiling information from DR_HOOK can go. A useful tool.
- ‘WALLTIME’ # another standard PBS or batch system setting. To paraphrase Einstein, this should be as large as necessary but no larger. There are different WALLTIMES set in this script, for different kinds of jobs (scalar and parallel; forecast and post-processing).

The various kinds of jobs here (“scalar”, “background” and “parallel”) are launched in different ways: either on the front-end node, or via the batch system to a single or multiple back-end nodes.

3.5 The “sms/config_exp.h” File

Once the more general details of your particular computer system and Harmonie data directories have been specified, this file is used to control each individual model run. Most likely you will need to edit a copy of this file for each separate Harmonie job that you submit. However, it can make sense to edit the original copy too, to specify those settings that you think will only rarely, if ever, change from run to run.

The key settings in this file include:

```
BUILD=<yesIno>    #  Once your original reference build has completed, you
                  #  can say “no” here.  Only say “yes” if you have changed
                  # some source code or are prepared to wait for the
                  # lengthy build process to take place all over again.

BINDIR= ${BINDIR-$HM_DATA/bin}      # Directory where Harmonie binary
```

```
                # files may be found after being built.  If not specified
                # externally by a pre-existing $BINDIR, this macro
                # specifies $HM_DATA/bin.  I found this did not work for
                # me; however, changing to $HM_DATA/lib/src did.
                # If $BINDIR doesn't exist already, Harmonie will not
                # create it itself.  This macro can cause some trouble!
                # See too the "bindir" section in util/makeup/main.mk .  I
                # find this section is not executed automatically, probably
                # because my $BINDIR is wrong, so I need to "mv"
                # ioassign and ODBTOOLS by hand.  Externally setting a
                # proper $BINDIR would probably fix all these problems.

DOMAIN=<IRELAND55…>     #  Various pre-set domain configurations
                # are specified further down in the config_exp.h file.  It is
                # important to specify only a domain for which you have
                # compatible input data.

LL=${LL-06}     # Forecast length in hours.  Normally you specify this
                # externally (from the "Harmonie start…" command line).

DFI="none"      # Digital filter initialization, a way of smoothing the
                # assimilation of boundary conditions.  Only use this if
                # 3DVAR assimilation is used (via ANAATMO, below).
                # No need for DFI otherwise.

ANAATMO=blending        # This macro specifies the kind of data
                # assimilation to be used.   The 3DVAR and 4DVAR
                # options may not be available on your system.  The "
                # blending" option means that initial conditions for any
                # forecast run should be "blended" with output from a
                # previous run (see FCINT below).  This presumes that a
                # previous run has been done, and that such output is
                # available.  This will typically NOT be the case initially,
                # and so you will need to set ANAATMO to "none", for a
                # "cold start" run.

ANASURF=CANARI_OI_MAIN  # See too the other options listed.  There
                # are dependencies between ANAATMO and ANASURF.
                # If ANAATMO=none, then ANASURF should be set to
                # "none" too.

SST=BOUNDARY    # Seems to be the right choice in most situations.

FCINT=06        # "Assimilation cycle interval".   This also relates to
                # ANAATMO.  Every $FCINT forecast hours, a Harmonie
                # run will look for output from a previous run to assimilate
                # into the current run – unless ANAATMO=none.  Where
                # DTGEND is specified at run-time, FCINT also specifies
```

```
                    # the interval between starting new forecast runs.

OBDIR=<path>        # Path to observations directory.  This is likely to be very
                    # system-specific.  Observations are needed to initialize
                    # most, if not all, runs.

HOST_MODEL="hir"        # Hirlam is the "host" or outer nesting model here.

BD_METHOD=gl_only       # gl_only probably better than the "old" method…

BDINT=<1|3|6>       # The interval (in hours) between updates of model
                    # boundary conditions (from external files).   Worth
                    # testing 1, 2, 3, or 6 hrs. here.

BDLIB=ECMWF         # BDLIB is irrelevant for non-ECMWF systems – even
                    # though it appears in other macros later in this file.  If
                    # "ECMWF" starts appearing in path-names of files that
                    # are "not found", you have some other problem!

BDDIR=/ichec/work/hirlam/hl_data_l10/@YYYY@@MM@@DD@_@HH@
                    # Very important directory containing (input) boundary
                    # data files, which are most likely output from earlier
                    # forecast model runs.  Note the date/time format – it can
                    # be subtly different from that in the default config_exp.h
                    # file.

INT_BDFILE=$WRK/ELSCF${CNMEXP}ALBC@NNN@
INT_SINI_FILE=$WRK/SURFXINI.lfi
                    # No need to change these macros.  They simply name
                    # temporary interpolated boundary files that show up in
                    # the run-time working directory and are removed at the
                    # end of each run.  You may notice the files
                    # (ELSCFHARMALBC*) during a run (or left behind after
                    # a failed run).

BDSTRATEGY=same_forecast    # No need to change this.

ARCHIVE=$HM_DATA/archive/$YY/$MM/$DD/$HH/  # This and the other
                    # "archive" macros are fairly self-explanatory.  No need to
                    # change them.

CREATE_CLIMATE=${CREATE_CLIMATE-yes}        # A "yes" here does not
                    # mean to run climate file generation from Harmonie
                    # output.  Rather, it means to take the generic climate
                    # data from HM_CLDATA (defined in Env_system) and
                    # convert it to a format suitable for use as input by the
                    # current Harmonie experiment.   So a "yes" means
                    # "generate run-specific climate data, for input to forecast
```

run, from generic climate files".

CLIMDIR=$HM_DATA/climate # Directory to place the run-specific input
# climate files, as pre-processed from the generic climate
# files in $HM_CLDATA.

BDCLIM=$HM_DATA/${BDLIB}/climate # This should be irrelevant to our
# Harmonie runs; seems it's only necessary for
# intermediate Aladin climate files. At least, the path
# through $BDLIB does not seem to be invoked at run-
# time at all.

OPERATIONAL_POSTP=yes # "Yes" here means post-process on the fly, as the model generates output every hour or so, rather than wait till the end of the run and post-process separately.

POSTP=yes # Yes or no, depending on whether you want "full" post-
# processing or not. "Yes" *should* work, so I use that.

PPTYPE="ie" # "ie" here is specific to the Irish Harmonie branch (see
# the Postpp script). It interpolates output fields to
# synoptic (standard) levels, among other things.

MAKEGRIB=yes # Yes, if you want to convert model output from native
# "fa" ("fichier arpege") format to "standard" grib format.

TOGRIB="his "$PPTYPE # Note the file prefixes explained in config_exp.h
# itself.

VERIFY=no # Only "yes" if you want to run verification (i.e., some kind
# of comparison with observations). Observations here
# seem to be raw "station" data – not processed,
# interpolated, gridded observations.

OBSEXTR=none # This is one place where BUFR files come in, if "yes"
# (the "extraction" seems to be of station data from the
# BUFR files).

FLDEXTR=yes # "Field" verification is different to "point-wise" (station by
# station) verification.

OBSMONITOR=yes # Sure why not?

4.0 Building Harmonie

After the Harmonie kit has been downloaded and the various configuration files edited, as outlined above, it is time to compile and link the various

Harmonie executables. This process is described well at https://hirlam.org/trac/wiki/Build_with_makeup .

Briefly, first make sure that your environment points to the compiler and MPI installations that are consistent with the settings in util/makeup/config.<your-compcentre-suffix>.

Next, if $HARMONIE_SRC is the top-level Harmonie src directory, and $HARMONIE_MAKEUP is the util/makeup directory, enter something like:

```
cd $HARMONIE_SRC
$HARMONIE_MAKEUP/configure -c -H -G -B -E sources.linux   \
   $HARMONIE_MAKEUP/config.ICHEC.stokes
gmake
```

All going well, this should build a large collection of libraries and executables in that same src directory. The “-c”, “-H” and other options each specify the building of different libraries. If you leave any of these out, you may end up with “unresolved” functions at link-time. Try “`$HARMONIE_MAKEUP/configure`” (with no arguments at all) for usage information.

The inverse of this situation can also arise. In the MAKEUP system, the configure script gathers and builds all the sources without knowing whether a given function or subroutine will be needed. Hence it cannot tell whether symbols missing from parts of the source are real problems or not: they may be functions called by routines that are not used in the current code. So, "gen_libdummy" is used to generate dummy subroutines for missing parts. This means that Harmonie can build successfully even if some libraries are missing; dummy subroutines are substituted for missing parts (depending on the makeup/config.* files).

So if you see warnings of the form: "Error: Should not be calling the dummy routine ", check the “undef” file in the source directory (e.g., $HM_DATA/$EXP/ICHEC.stokes/src/undef) to see the complete list of undefined symbols.

The “configure” command above essentially makes the Makefiles for each Harmonie component, while the “gmake” (or “make”) command invokes all those Makefiles to actually do the compiling and linking. So if you change anything in $HARMONIE_MAKEUP/config.<your-compcentre-suffix>, you will need to re-run “configure” before re-running “gmake”.

Individual sections can be built (or “cleaned”) separately, using, e.g.:

```
gmake PROJ=xrd
gmake PROJ=arp clean
```

5.0 Running Harmonie

Once the executables are all built, a Harmonie run, or "experiment" can be prepared and started. There is a good tutorial on how to do this at: https://hirlam.org/trac/wiki/HarmonieSystemDocumentation/Harmonie-mSMS

Briefly, put <path-to-Harmonie>/config-sh on your $PATH (so that the Harmonie script there can be found); create an experiment directory under $HOME/hm_home, and run:

```
Harmonie setup –r <path-to-Harmonie> -h <your-config-suffix>
```

Formally, the arguments to "Harmonie setup" are:

```
Harmonie setup -r <release> -h <host> -c <config>
```

Where <host> is the suffix to your config.* files (e.g., ICHEC.stokes), and <config> is one of the pre-cooked experiments defined in Harmonie_testbed.pl.

"Harmonie setup" should take less than a second to complete, and just places some configuration files in your current directory. There are 3 main files there:

- Env_system, a soft-link to config-sh/config.<your-compcentre-suffix>, as outlined in section 3.3 above;
- Env_submit, a soft-link to config-sh/submit.<your-compcentre-sufix>, as outlined in section 3.4 above;
- sms/config_exp.h , a copy of the sms/config_exp.h file from the Harmonie reference installation, as outlined in section 3.5 above.

Each of these files may need further editing in this "experiment" directory to control whatever experiment you want to run. E.g., the batch-system WALLTIME or number of compute nodes may need to be changed in Env_submit; some input data directories may need to be changed in Env_system, and almost certainly, the particular experimental configuration will need to be set in sms/config_exp.h .

The preparation of those 3 files is what consumes most human-time. Once they are ready, the computers can finally be put to work. That is done with the basic command "Harmonie start", as in, e.g.,

```
Harmonie start EXP=<name> DTG=yyyymmddhh DTGEND=yyyymmddhh LL=hh
```

Here:

- **EXP** is the name of the experiment. It defaults to the name of the current directory if not explicitly specified.
  (Note that EXP names containing dashes ("-") seem to be not allowed; use underscores instead. E.g., "EXP=met-omp-55" provoked the error message: "met-omp-55 is not a legal node name", while

"EXP=metomp_55" worked perfectly well.)

- **DTG** is the start-time for the experiment in the year-month-day-hour format shown.
- **DTGEND** is the start-time for the last model run within the current suite of overlapping forecasts. Remember that FCINT (in config_exp.h) specifies an interval (usually 6 hours) after which a new forecast integration started. New forecasts continue to be started every FCINT hours until DTGEND.
- **LL** is the length of each forecast run in hours. It defaults to 6.

5.1 Monitoring (and debugging) Harmonie runs.

If you are lucky, the experiment you started will run successfully to completion. You will probably discover to your surprise, however, that Harmonie runs can sometimes fail. So here are some pointers to where to look for explanations of where it all went wrong.

Each experiment is run under $HM_DATA (i.e., probably the same as $SCRATCH/hm_data/$EXP). In there, overall experiment progress can be followed in the various log-files that appear (mSMS.log, harmonie.log, etc.).

The progress of each component of Harmonie also can be followed (following the mXCdp graphical display) in the various directories that appear under $EXP (i.e, $HM_DATA/$EXP). E.g., the first set of component files to appear are:

```
InitRun.1  InitRun.job1  InitRun.job1-q
```

Here, the "*.job1" file is a shell script to run the component; "*.job1-q" is that shell script wrapped up as a batch-submission script (mainly containing extra batch-system related header information at the top); and the "*.1" file is the output log. Look in there for information about job progress or job failure.

5.2 Modifying Individual Files (e.g., Sources or Scripts)

Local or individual file edits should be done in the $HM_HOME/$EXP directories, not in the "reference" source directories, but using the same directory structure as the reference set. Files can be "checked out" with, e.g.,

```
Harmonie co scr/RunCanari
Vi scr/RunCanari
```

If source files are changed like this, be sure to allow them to be re-built before running, by setting "BUILD=yes" in sms/config_ex.h .

6.0 Harmonie Data Files

Here is a partial list of the various Harmonie binary data files that can appear during and/or after each experiment run. Here I assumed that $CNMEXP is set to "HARM" (in config_exp.h). Files of type 2 appear in $OBDIR, type 3-4 in $BDDIR, while types 5-8 appear in $HM_DATA/archive/YYYY/MM/DD/HH:

1. mMM: "climate" file names, for MM from 01 through 12 (January – December). These are generated from generic climate data files in HM_CLDATA and placed in CLIMDIR, ready to be used as input to $EXP. Typically, only 3 such files are needed for a short Harmonie run, corresponding to the 3 months spanning the current date.
2. obYYYYMMDDHHHH: observational data at date/time specified, in grib (?) format. Located in $OBDIR, and used during data assimilation as input for Harmonie; also, for post-processing verification.
3. anYYYYMMDD_HH+hhh: analysis (assimilated, gridded, pre-processed) fields at date/time specified, for forecast hours (hhh) specified. Located in $BDDIR, used as boundary input data for Harmonie.
4. fcYYYYMMDD_HH+hhh: forecast fields from run started at date/time specified, run for hhh hours. Located in $BDDIR, used as boundary data (input) to Harmonie.
5. AROMOUT_hhhh.lfi : Arome physics output after hhhh forecast hours, in FA format.
6. ICMSHHARM+hhhh : Harmonie output after hhhh forecast hours, in FA format.
7. PFHARM000+hhhh : Harmonie surface (?) output after hhhh forecast hours, in FA format.
8. fcYYYYMMDDHH+hhhgrib : Output in grib format, for run started YYYYMMDDHH, and forecast for hhh hours.
9. ELSCFHARMALBCnnn : intermediate (interpolated) boundary file. These show up in the run-time directory yyyymmdd_hh, and are linked to by bdinput_hh files (unless no boundary data is available for needed bdinput_hh file, in which case a "fc" forecast file is used instead. The nnn index just increases sequentially – not to be interpreted as a time or hour value.
10. Bdinput_hh: boundary input files, presumably. The hourly values hh come from the BDINT parameter set in sms/config_exp.h. (Typically 1-3 hours). These are soft-links to ELSCFHARMALBCnnn (if such a boundary file exists), or to "fc*' forecast output if not.
11. Vfld${EXP}YYYYMMDDHHhh : verification fields from run started at date/time specified after hh hours forecast. Ascii file containing pseudo-station data (this file has plenty of "-99.00" values, i.e., missing data of some kind…), presumably for comparison ("validation") with real observational data.

7.0 Further information

Other topics of interest include:

- **SMS** (or mini-SMS, mSMS, as used outside ECMWF). Stands for "Supervisor Monitor Scheduler". SMS is aware of the dependencies between the various parts of a Harmonie experiment, and knows when it is safe to submit each part to the queue. It interacts with xMCdp. See http://www.ecmwf.int/products/data/software/sms.html .
- **xMCdp** (xMotif Command and Display program). A graphical display of a Harmonie experiment's progress. Users can interact with SMS via xMCdp.
- **Xgrbplt** (or other tool): Graphical display of model output.

8.0 Compiler Macro Settings

Some macros that might be present in the util/makeup/config.<compcentre> file:

- -DNO_CURSES : no need to use this on systems where the curses library does exist (see e.g., /usr/include/curses.h)
- -DUSE_SAMIO : use this for asynchronous (parallel) IO. See, e.g., xrd/module/samil_mod.F90. If this macro is set, it is also necessary to specify the environment variable SAMIO_NUM_IOPES, and ensure that the number of allocated cores is at least as big as $NPROC (as in mpirun –np $NPROC) *plus* $SAMIO_NUM_IOPES.
- -DHIGHRES : Leave this set. It seems to be used only in uti/pinuts/module/const_standart_mod.F90, where if on, it sets MAXLAT, MAXLEV, and MXTRO to 1601, 999, and 800, resp. (instead of default lower values).

Enda O'Brien
enda.obrien@ichec.ie