# New modular compiling and testing environment for OASIS3-MCT

Laure Coquart, S. Valcke, Anthony Craig

# *New modular compiling and testing environment for OASIS3-MCT*

**Coquart L., Valcke S., Craig A.**

**CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France**

**Technical Report TR-CMGC-23-175**

This work was carried out in the framework of the
EU project H2020 IS-ENES3 number 824084.



## Table des matières

# 1 Introduction

OASIS3-MCT is a numerical multi-component coupler widely used in the climate community, developed at Cerfacs in collaboration with CNRS (Craig, A. et al. (2017)).

The different OASIS3-MCT versions are managed by GIT on the Cerfacs nitrox server since March 2019 and the developments were tested and validated using Buildbot since 2012. In 2021 all test cases were rewritten. In this latest test suite, the different test cases, called "toys" hereafter, consist usually of two component models, not containing any physics but invoking functionalities of the coupler and reproducing coupling algorithms of real coupled models (Coquart, L. et al. (2021)).

A drawback of this test suite under Buildbot was that, due to its relative complexity, only one person could run the Buildbot tests and analyze the results. It was also difficult to run only one test for one toy. This is the reason why a more modular environment of compilation and run, using toys based on the Buildbot system was created and is available on OASIS3-MCT master branch since December 2022 (commit 5e9efd18).

In the new environment, all the data needed to run a toy as well as the parameter files are located in the toy directory. This was not the case before when using Buildbot.

Defining only two global environment variables, it is now possible to compile and run different tests for each toy after having compiled OASIS3-MCT "manually". Top-level scripts were also developed to chain one or more tests for one or more toys creating the ability to run a test suite.

The first part of the document focuses on the two global environment variables that drive compilation of OASIS3-MCT and compilation and run of the different toys. The second section of the document presents the top-level scripts to compile OASIS3-MCT and the toys and then to run them chaining them automatically. The third part summarizes how to create, compile and run a new toy on a new computer using local scripts and using top-level scripts. Then we provide conclusions in the final section.

# 2 Compiling and running the toys in the new environment using two global environment variables

## 2.1 Compiling OASIS3-MCT "manually"

As usual, OASIS3-MCT must first be compiled (Valcke, S. et al. (2021)). In the new modular compiling environment of OASIS3-MCT, two global environment variables must be defined where the compilation occurs:

- **OASIS_COUPLE** : the location of the sources of the OASIS3-MCT coupler (/lib, /util, /examples, etc. directories)
- **OASIS_ENV** : The user-defined name specifying the machine, compiler, and overall environment. This defines the comp_env and header Makefile files.

A header Makefile, named make.${OASIS_ENV}, adapted to the platform where OASIS3-MCT is compiled, must be located in the directory oasis3-mct/util/make_dir. We recommend to compose ${OASIS_ENV} as something like (computer_name)_(compiler_version)_(MPI_library_version) [for example, tioman_intel21.1.1_intelmpi2021.1.1], but any unique string defined by the user is acceptable. The header Makefile defined by make.${OASIS_ENV} is automatically included in the OASIS3-MCT TopMakefileOasis3 Makefile via the generic make.inc file. This is shown in Appendix A section 7.1.

A file named comp_env_${OASIS_ENV}.sh must also exist in the directory oasis3-mct/util/make_dir. It sets up the computer environment for compiling and running the tests via module or other system calls. This file is automatically executed by the test scripts. The contents of the sample file comp_env_tioman_intel21.1.1_intelmpi2021.1.1.sh used for a Linux fedora 26 computer called tioman (compute_name) with intel21.1.1 (compiler_version) and intelmpi2021.1.1 (MPI_library_version) is shown in Appendix A section 7.2 and available in directory oasis3-mct/util/env_tests .

Examples of make.${OASIS_ENV} and comp_env_${OASIS_ENV}.sh are provided in directory oasis3-mct/util/make_dir as shown at Figure *1***Erreur ! Source du renvoi introuvable.**:

```
coquart@tioman: /space/coquart/oasis3-mct/util/make_dir [16:58:49]  (master)
 $ --> ls
comp_env_belenos_intel2018.5.274_intelmpi2018.5.274.sh  make.common
comp_env_fundy_gfortran10.2.0_openmpi4.1.0.sh           make.fundy_gfortran10.2.0_openmpi4.1.0
comp_env_kraken_intel18.0.1.163_intelmpi2018.1.163.sh   make.inc
comp_env_scylla_intel2018.4.274_intelmpi2018.4.274.sh   make.kraken_intel18.0.1.163_intelmpi2018.1.163
comp_env_stiff_pgi20.4_openmpi3.1.3.sh                  make.mac_gfortran10_openmpi4.0.5
comp_env_tioman_intel21.1.1_intelmpi2021.1.1.sh         make.scylla_intel2018.4.274_intelmpi2018.4.274
header_examples                                         make.stiff_pgi20.4_openmpi3.1.3
make.belenos_intel2018.5.274_intelmpi2018.5.274         make.tioman_intel21.1.1_intelmpi2021.1.1
make.cerfacs_gfortran7.3.1_openmpi4.0.5                 TopMakefileOasis3
```

Figure 1: Files in repository oasis3-mct/util/make_dir

The compilation of OASIS3-MCT sources can be done in directory oasis3-mct/util/make_dir by typing "make -f TopMakefileOasis3" (see section 6.1 of the User Guide for details).

## 2.2 Compiling and running the toys using local scripts

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Quick Start Example**: Compile and run tutorial_communication toy on Linux CECI-Cerfacs computer tioman

cd /space /coquart/
git clone https://gitlab.com/cerfacs/oasis3-mct.git
export OASIS_COUPLE = /space/coquart/oasis3-mct
export OASIS_ENV = tioman_intel21.1.1_intelmpi2021.1.1
cd ${OASIS_COUPLE}/util/make_dir
make -f TopMakefileOasis3
cd ${OASIS_COUPLE}/examples/tutorial_communication
ln -s ${OASIS_COUPLE}/env_tests/oasis_test_build.sh .
ln -s ${OASIS_COUPLE}/env_tests/oasis_tes_run.sh .
Define one test to do in a file called param_tutorial_test01, param_tutorial_test02 …
Compile the toy :
./oasis_test_build.sh param_tutorial_communication_test01
Run the toy :
./oasis_test_run.sh param_tutorial_communication_test01
Output is in:
${OASIS_COUPLE}/examples/tutorial_communication/TESTS/work_tutorial_communication_namcouple_Makefile_atmos_1_ocean_1

All files and parameters used above are described below.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

The toy model examples in oasis3-mct/examples have been adapted to the new compiling and running environment (except for running /regrid_environment and /spoc/spoc_regridding, as these are especially complex).

In addition, the toys of the nitrox project “oasis3-mct_tests”[1] have also been adapted to this new environment since 2021; the list of toys in oasis3-mct_tests is given in Figure 2:

```
coquart@tioman: /space/coquart/oasis3-mct_tests [19:55:30]  (master)
 $ --> ls
clean_files_toys.sh                 toy_grids_regional_to_regional  toy_multiple_fields_one_communication
toy_1f1grd_to_2f2grd                toy_grids_writing               toy_multiple_grids_per_partition
toy_auxiliary_routines              toy_identical_grids             toy_NLOGPRT
toy_bundle                          toy_interpolation               toy_NMATXRD_options
toy_CHECKIN_BLASOLD_BLASNEW_CHECKOUT toy_interpolation_field        toy_NTHRESH_STHRESH
toy_configuration_components_A      toy_intracomm                   toy_restart_ACCUMUL_1_LAG
toy_configuration_components_ABCGH  toy_load_balancing              toy_restart_ACCUMUL_1_NOLAG
toy_configuration_components_B      toy_mapcons                     toy_restart_ACCUMUL_2_LAG
toy_configuration_components_C      toy_maphot_1field               toy_restart_ACCUMUL_2_NOLAG
toy_configuration_components_CGH    toy_maphot_2field               toy_runoff
toy_create_couplcomm                toy_MAPPING_options             toy_scalar_coupling
toy_gaussianreducedgrid             toy_mixed_SP_DP                 toy_time_transformations
```

Figure 2: Toys from oasis3-mct_tests adapted to the new modular environment

Figure 3 illustrates the ksh scripts oasis_test_build.sh and oasis_test_run.sh available in oasis3-mct/util/env_tests and used to compile and run the toy.

[1] One can get this test suite writing to oasishelp@cerfacs.fr

A parameter file containing user-defined parameters and named param_${casename}, where ${casename} is the name of the toy directory, must be provided in the toy directory; as multiple tests can be done using one single toy, there can be multiple parameter files named param_${casename}_test01, param_${casename}_test02, etc. This structure of the parameter file name is mandatory. Each file specifies:

- the name of the executables, USER_EXE1 until USER_EXE5 if it is necessary,
- the number of processors for each model, USER_NPEXE1 until USER_NPEXE5 if it is necessary,
- the name and directory of the Makefile, USER_MAKEFILE and USER_MAKELOC
- the name and directory of the namcouple, USER_NAMCOUPLE and USER_NAMLOC
- the remapping file directory, USER_RMPLOC,
- the directory containing OASIS3-MCT auxiliary files grids.nc, areas.nc and masks.nc, USER_AUXLOC,
- the directory containing the coupling restart files, USER_RSTLOC
- the directory that contains the file defining the model grid, USER_MESHLOC.

Specifying all these parameters is mandatory, except that if there is only one executable, it is necessary to define only USER_EXE1 and USER_NPEXE1.
An example of a parameter file for examples/tutorial_communication is given appendix A section 7.37.3

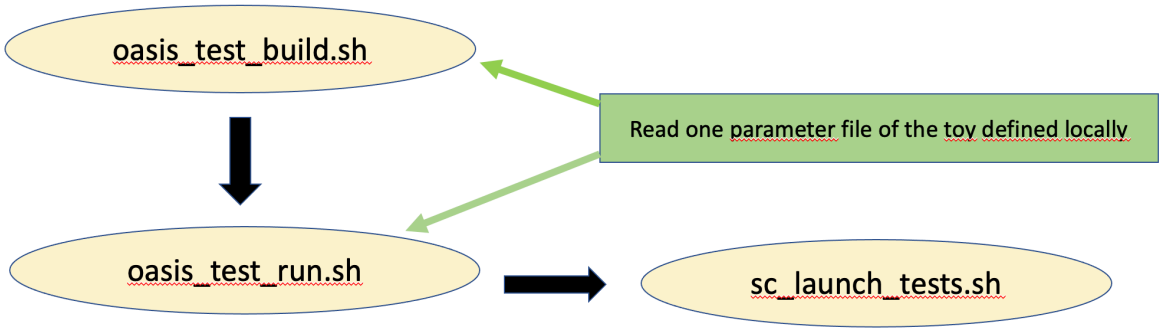


Figure 3: Scripts used to locally compile and run the toys of the new modular environment of OASIS3-MCT tests

Thanks to the definition of the two environment variables **OASIS_COUPLE** and **OASIS_ENV** (see section 2.1), the script oasis_test_build.sh **is able** to compile the toy. It has to be called with the parameter file as argument, e.g.:
./ oasis_test_build.sh param_tutorial_communication_test01
It sources the script ${OASIS_COUPLE}/util/make_dir/comp_env_${OASIS_ENV}.sh, defining the computer environment and the parameter file, copies the Makefile locally and uses it to compile the sources of the toy. Note that the Makefile to compile the toy sources has to start with the line
include $(OASIS_COUPLE)/util/make_dir/make.inc
The script oasis_test_build.sh is described in Appendix A section 7.4.

Then the script oasis_test_run.sh, reading the parameter file of the toy, can be used to run one test with the toy on a specific computer. It has to be called with the parameter file as an argument, e.g.:

./ oasis_test_run.sh param_tutorial_communication_test01
It sources the script ${OASIS_COUPLE}/util/make_dir/comp_env_${OASIS_ENV}.sh, defining the computer environment, and the parameter file, and calls the script ${OASIS_COUPLE}/util/env_tests/sc_launch_tests.sh that launches the test. The results are stored in a local repository called TESTS in the toy repository. The file oasis_test_run.sh is described in Appendix A section 7.5.

To be able to compile OASIS3-MCT and run many tests for many toys on many platforms, some top-level scripts were developed (see scripts in oasis3-mct/util/env_tests). They are described in the next section.

# 3 Compiling and running the toys using top-level scripts

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
**Quick Start Example**: Compile and run tutorial_communication on Linux CECI-Cerfacs computer tioman

Once the toy tutorial_communication has been adapted to the new compiling and running environment, one can use the top-level scripts to compile and run it:

export OASIS_COUPLE = /space/coquart/oasis3-mct

export OASIS_ENV = tioman_intel21.1.1_intelmpi2021.1.1

cd ${OASIS_COUPLE}/util/env_tests

Compile and run test suite defined in env_tests_param_example on Linux :

./scrip_top.sh example

Output is in:
${USER_RUNDIR}/work_tutorial_communication_namcouple_Makefile_atmos_1_ocean_1

All files and parameters used above are described below.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Instead of compiling and running the toys locally in their own directory, it is possible to use top-level scripts defined in oasis3-mct/util/env_tests.
A list of the files contained in oasis3-mct/util/env_tests is shown below Figure 4:

```
coquart@tioman: /space/coquart/oasis3-mct/util/env_tests [18:05:48]  (master)
 $ --> ls
env_tests_param_example                          oasis_test_build.sh  sc_compile_oasis.sh  sc_top.sh
env_tests_param_tioman_intel21.1.1_intelmpi2021.1.1  oasis_test_run.sh    sc_launch_tests.sh
```

Figure 4: Files in oasis3-mct/util/env_tests

Figure *5* describes the tasks sequencing to compile OASIS3-MCT and a selection of toys, and run these toys using different top-level scripts:

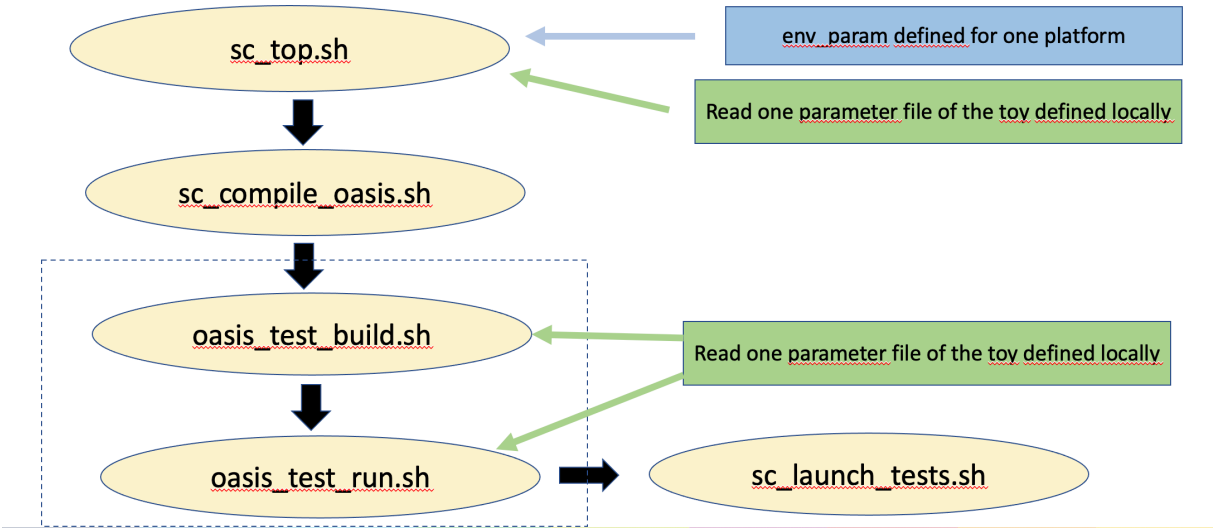


Figure 5: Tasks sequencing to compile and run a selection of toys in the new modular environment of OASIS3-MCT tests using top-level scripts

A file env_tests_param_xxxxx, where xxxxx is a string defined by the user, specifies the tests to be automatically compiled and run. That file must exist in the repository oasis3-mct/util/env_tests; it contains the name of the toys, the number of tests for each toy and the name of the parameter file for each toy. A very simple example of this file, env_tests_param_example, is given Appendix A section 7.6. In that example, the user specifies:

- to compile OASIS3-MCT (OASIS_COMPILE=TRUE)
- to compile and run two different toys (USER_TOY=("/scratch/globc/valcke/oasis3-mct/examples/tutorial_communication" "/scratch/globc/valcke/oasis3-mct/examples/spoc/spoc_communication"))
- for tutorial_communication, to run 2 different tests, and for spoc_communication, to run only one test (USER_TEST=("2" "1"));
- for tutorial_communication and for spoc_communication to take the parameter files from the directories specified USER_PARAMLOC , here the local toy directories; note that tutorial_communication will automatically take respectively param_tutorial_communication_test01 and param_tutorial_communication_test02 from /scratch/globc/valcke/oasis3-mct/examples/tutorial_communication as input parameter file, and spoc_communication will automatically take param_spoc_communication_test01 from /scratch/globc/valcke/oasis3-mct/examples/spoc/spoc_communication as input parameter file
- to run the tests in USER_RUNDIR=${OASIS_COUPLE}/OA3_MCT_RES

The user can then compile and run this series of tests with "./sc_top.sh example" and use env_tests_param_example as input file. (Note that the argument "example" will specify use of env_tests_param_example). The script sc_top.sh is described Appendix A, section 7.7. As shown in Figure 5, the script sc_top.sh then calls sc_compile_oasis.sh to compile OASIS3-MCT if the global environment variable ${OASIS_COMPILE} is set to TRUE. The script sc_compile_oasis.sh is given in appendix A section 7.8. Then the script sc_top.sh compiles and runs each toy and test specified by the user using the local scripts oasis_test_build.sh and oasis_test_run.sh in each toy directory.

Some additional variables are defined by default but they can be specified in env_tests_param_xxxxxx if necessary, see for example the file env_tests_param_tioman_intel21.1.1_intelmpi2021.1.1 given in Appendix A section 7.9 and available in directory oasis3-mct/util/env_tests.
In this file, the user may define:

- OASIS_ROOT to specify the location of the OASIS3-MCT sources. It is useful when changing the computer where the tests are done
- OASIS_COUPLE to avoid defining it "by hand" as an environment variable (see 2.1). It is also useful when testing OASIS3-MCT on different platforms
- OASIS_ENV to avoid defining it "by hand" as an environment variable (see 2.1). It is also useful when testing OASIS3-MCT on different platforms
- OASIS_TESTS for the location of the toys. It is useful when changing of computer where the tests are done
- OASIS_COMPILE=TRUE; OASIS_DEBUG=TRUE and OPENMP=TRUE: to compile OASIS, with debugging options, and open MP options
- USER_TOY to define the list of toys to compile and run

- USER_TEST to define the number of tests for each toy
- USER_PARAMLOC to specify the localization of the parameter files
- USER_RUNDIR to define where to run the tests

# 4 Summary

In summary, to create, run, and compile a new toy in this environment on a new computer using specific compiler and mpi libraries, one must first compile OASIS3-MCT in /oasis3-mct/util/make_dir on the new computer. For that, it is necessary to define the environment variables OASIS_COUPLE, the location of the OASIS3-MCT sources, and OASIS_ENV, the name of the computer environment. The user then has to generate the header Makefile corresponding to the new computer, make.${OASIS_ENV} and the computer environment file, comp_env_${OASIS_ENV}.sh. The computer environment file will be used to compile and run the toy(s) on the new computer. The header Makefile is automatically included in the file make.inc (see 7.1) which is included in the TopMakefileOasis3 file that compiles OASIS3-MCT. We recommend OASIS_ENV to be composed like (computer_name)_(compiler_version)_(MPI_library_version). There are some examples of these files in /oasis3-mct/util/make_dir.

Then, a toy test case can be created anywhere on the computer. The first step should be to copy or link the scripts oasis_test_build.sh and oasis_test_run.sh from oasis3-mct/util/env_tests files in the toy directory (see Figure *3*: Scripts used to locally compile and run the toys of the new modular environment of OASIS3-MCT tests). These scripts use the environment variables OASIS_COUPLE and OASIS_ENV (see above). The local Makefile used to compile the toy sources has to start with the line
"include $(OASIS_COUPLE)/util/make_dir/make.inc".
The different tests to be performed with the toy must be configured in a local parameter file, named "parameter_toyname_test01", "parameter_toyname_test02", etc. … (see 7.3). The toy can then be compiled and run using the compile and run scripts with the parameter file given as argument, for example "**./oasis_test_build.sh parameter_toyname_01"** and "**./oasis_test_run.sh parameter_toyname_01"**.

One can also use the top-level script oasis3-mct/util/env_tests/sc_top.sh to compile and run different tests for different toys at once (see Figure *5*: Tasks sequencing to compile and run a selection of toys in the new modular environment of OASIS3-MCT tests using top-level scripts). In addition to the files described in the previous paragraph (to compile and run one test for one toy), one has to define the different tests for the different toys in a file env_tests_param_xxxxx in the directory oasis3-mct/util/env_tests where xxxxx is a string defined by the user. Some additional variables, giving more flexibility in the test setups, can also be defined in env_tests_param_xxxxx (see section 3 and Appendix 7.9). The tests can then be launched using the command "**./sc_top.sh xxxxx"**.

# 5 Conclusion

This report presents the new modular environment used to compile and run OASIS3-MCT tests. It is based on the definition of two global environment variables, **OASIS_COUPLE** and **OASIS_ENV**, and of two files linked to the platform, make.${OASIS_ENV} and comp_env_${OASIS_ENV}.sh, for compilation and for running the tests in the pre-defined environment.

Section 2 of the document describes how to compile OASIS3-MCT “manually” defining these two environment variables. Then with the definition of the same variables, it is possible to compile and run each toy locally in their own directory using the scripts oasis_test_build.sh and oasis_test_run.sh. The two scripts read a parameter file containing the details of the test to perform, and source the compiling environment comp_env_${OASIS_ENV}.sh. The script oasis_test_run.sh calls the script sc_launch_tests.sh to run the test in the corresponding platform environment.

Section 3 describes the scripts that can be used to compile and run a series of toys in the new modular environment using the top-level script sc_top.sh. It is also based on the two global environment variables **OASIS_COUPLE** and **OASIS_ENV** and on the local scripts oasis_test_build.sh and oasis_test_run.sh. But sc_top.sh allows to compile OASIS3-MCT and compile and run a series of different tests for different toys based on a user-defined configuration in the file env_tests_param_xxxxx.

Finally, section 4 provides a summary.

In a near future, this new modular environment will be run under Gitlab CD/CI (GitLab CI/CD (2023)) on the Cerfacs nitrox server to follow and debug the developments done in OASIS3-MCT under git, instead of using Buildbot.

# 6 Bibliography

GitLab CI/CD tool for software development (2023) *https://docs.gitlab.com/ee/ci/*

Coquart, L., Valcke, S., Craig, A. and Maisonnave, E. (2021) *New Buildbot test suite for the OASIS3-MCT coupler Fortran source code*, CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France, TR-CMGC-21-36, Technical report

Valcke, S., Craig, A., Maisonnave, E. and Coquart, L. (2021) *OASIS3-MCT User Guide, OASIS3-MCT 5.0*, CECI, Université de Toulouse, CNRS, CERFACS, Toulouse, France - TR-CMGC-21-161, Technical report

Craig, A., Valcke, S. and Coquart, L. (2017) *Development and performance of a new version of the OASIS coupler, OASIS3-MCT 3.0*, Geoscientific Model Development, 10, pp. 3297-3308, doi: 10.5194/gmd-10-3297-2017

# 7 Appendix A

## 7.1 File make.inc in oasis3-mct/util/make_dir

*include $(OASIS_COUPLE)/util/make_dir/make.$(OASIS_ENV)*

## 7.2 One example of a comp_env_${OASIS_ENV} file: comp_env_tioman_intel21.1.1_intelmpi2021.1.1.sh

This file comp_env_tioman_intel21.1.1_intelmpi2021.1.1.sh is used to set up the environment of compilation and run on Fedora 26 tioman computer.

```
#!/bin/ksh
############################
# Compilation environment
############################
source /etc/profile.d/modules.sh
module purge
module load intel/21.1.1
module load intelmpi/2021.1.1
module load lib/netcdf-fortran/4.4.4_phdf5_1.10.4
module load python/3.7.7
echo 'We work on tioman'
echo `which mpirun`
export MPIRUN=mpirun
export corespn=1
```

## 7.3 One example of a toy parameter file: param_tutorial_communication_test01

```
#########################################################
## PARAMETERS for test 1 for toy tutorial_communication
## By default, OASIS_TOYDIR is defined as `pwd` in oasis_test_build.sh and oasis_test_run.sh;
## If top-level script sc_top.sh is used to run multiple tests, OASIS_TOYDIR is defined in sc_top.sh
#########################################################
export USER_EXE1=atmos
export USER_EXE2=ocean
export USER_NPEXE1=1
export USER_NPEXE2=1
export USER_MAKEFILE=Makefile
export USER_MAKELOC=${OASIS_TOYDIR}
export USER_NAMCOUPLE=namcouple
export USER_NAMLOC=${OASIS_TOYDIR}/data_tutorial
export USER_RMPLOC=${OASIS_TOYDIR}/data_tutorial
export USER_AUXLOC=""
export USER_RSTLOC=${OASIS_TOYDIR}/data_tutorial
export USER_MESHLOC=${OASIS_TOYDIR}/data_tutorial
```

## 7.4 Ksh script oasis_test_build.sh (to compile a toy)

```
#!/bin/ksh
################################################
# USAGE:
# Define following the variable environment:
# OASIS_COUPLE: the location of the sources of the coupler (/lib, /util, /examples, etc. dire
ctories)
# OASIS_ENV: the extension of the header Makefile to use for OASIS3-MCT compilation
#
# Then for example
# ./oasis_test_build.sh param_casename_test01
# ./oasis_test_build.sh param_casename_test02
# ...
#
# The param_casename_test?? are files that exists in the toy
# directory and specify several aspects of each test run
#
# Be carefull that OASIS3-MCT is compiled with the same environment than the toy
################################################

testname=$1
echo "testname = $testname"
if [ ! -f ./$testname ]; then
  echo "ERROR in param file argument, usage"
  echo "  ./oasis_test_build.sh \$testname"
  exit -9
fi

srcdir=`pwd`
export casename=`basename $srcdir`
export pathname=`dirname $srcdir`

if [ -z "${OASIS_ENV}" ]; then
  echo "ERROR OASIS_ENV not defined"
  exit -9
fi

if [ -z "${OASIS_COUPLE}" ]; then
  echo "ERROR OASIS_COUPLE not defined"
  exit -9
fi

if [ -z "${OASIS_TOYDIR}" ]; then
  export OASIS_TOYDIR=`pwd`
  echo "OASIS_TOYDIR is by default ${OASIS_TOYDIR}"
else
  echo "OASIS_TOYDIR is set in sc_top.sh and is ${OASIS_TOYDIR}"
fi

. ${OASIS_COUPLE}/util/make_dir/comp_env_${OASIS_ENV}.sh
```

```
. ./$testname
cp -f ${USER_MAKELOC}/${USER_MAKEFILE} ${OASIS_TOYDIR}/Makefile
make clean
make
```

## 7.5 Ksh script oasis_test_run.sh (to run a test)

```
#!/bin/ksh
#set -xv
################################################
# USAGE:
# Define following the variable environment:
# OASIS_COUPLE: the location of the sources of the coupler (/lib, /util, /examples, etc. directories)
# OASIS_ENV: the extension of the header Makefile to use for OASIS3-MCT compilation
#
# Then for example
# ./oasis_test_run.sh param_casename_test01
# ./oasis_test_run.sh param_casename_test02
# ...
# The param_casename_test?? are files that exists in the toy
# directory and specify several aspects of each test run
#
################################################

testname=$1

echo "testname = $testname"
if [ ! -f ./$testname ]; then
  echo "ERROR in param file argument, usage"
  echo "  ./oasis_test_run.sh \$testname"
  exit -9
fi

if [ -z "${OASIS_ENV}" ]; then
  echo "ERROR OASIS_ENV not defined"
  exit -9
fi

if [ -z "${OASIS_COUPLE}" ]; then
  echo "ERROR OASIS_COUPLE not defined"
  exit -9
fi

srcdir=`pwd`
export casename=`basename $srcdir`
export pathname=`dirname $srcdir`

if [ -z "${OASIS_TOYDIR}" ]; then
  export OASIS_TOYDIR=`pwd`
  echo "OASIS_TOYDIR is by default ${OASIS_TOYDIR}"
```

```
else
  echo "OASIS_TOYDIR is set in sc_top.sh and is ${OASIS_TOYDIR}"
fi

if [ -z "${USER_RUNDIR}" ]; then
  export USER_RUNDIR=${OASIS_TOYDIR}/TESTS
  echo "USER_RUNDIR is by default ${OASIS_TOYDIR}/TESTS"
else
  echo "USER_RUNDIR is set in env_param file and is ${USER_RUNDIR}"
fi

. ${OASIS_COUPLE}/util/make_dir/comp_env_${OASIS_ENV}.sh

# Need to reinitialize some variables
# to run toys with different models one after the other
# Do not modify below, use $testname
export USER_EXE1=
export USER_EXE2=
export USER_EXE3=
export USER_EXE4=
export USER_EXE5=
#########################################################
. ./$testname
${OASIS_COUPLE}/util/env_tests/sc_launch_tests.sh
```

## 7.6 One example of a of env_tests_param_xxxxx : env_tests_param_example (tutorial_communication and spoc_communication)

```
#############################################
## USER SECTION
#############################################
# OASIS VARIABLES DEFINITION
# + NAMES OF THE DIFFERENT TOYS
#############################################
# To recompile oasis or not
export OASIS_COMPILE=TRUE
# List of the toys to run (complete name including directory)
export USER_TOY=("/scratch/globc/valcke/oasis3-mct/examples/tutorial_communication"
"/scratch/globc/valcke/oasis3-mct/examples/spoc/spoc_communication")
# Number of tests for each toy
export USER_TEST=("2" "1")
# Localization of parameter files for each toy
export USER_PARAMLOC=("/scratch/globc/valcke/oasis3-mct/examples/tutorial_communication"
"/scratch/globc/valcke/oasis3-mct/examples/spoc/spoc_communication")
# Root directory for all tests
export USER_RUNDIR=${OASIS_COUPLE}/OA3_MCT_RES
```

## 7.7 Top-level ksh script sc_top.sh

```
#!/bin/ksh
#set -xv
################################################
################################################
# Link the correct env_tests_param
################################################
################################################
envcomp=$1
echo "envcomp=$envcomp"
if [ -z "$1" ]; then
   echo "No environment argument supplied"
   exit -9
fi
ln -sf env_tests_param_${envcomp} env_tests_param
. ./env_tests_param
################################################
################################################
# Compilation of OASIS3-MCT OR PYOASIS only once
# Creation of library verification
# comp_env_${OASIS_ENV}.sh and make.${OASIS_ENV}
# must exist
################################################
################################################
if [ -z "${OASIS_ENV}" ]; then
  echo "ERROR OASIS_ENV not defined"
  exit -9
fi

if [ -z "${OASIS_COUPLE}" ]; then
  echo "ERROR OASIS_COUPLE not defined"
  exit -9
fi
. ${OASIS_COUPLE}/util/make_dir/comp_env_${OASIS_ENV}.sh
#
if [ -z "${OASIS_COMPILE}" ]; then
  echo "BE CAREFULL OASIS_COMPILE not defined, OASIS will not be compiled alone"
elif [ ${OASIS_COMPILE} == TRUE ]; then
          echo "OASIS_COMPILE is set to TRUE, OASIS will be compiled alone"
          . ./sc_compile_oasis.sh
fi
################################################
################################################
# Loop over the toys:
# compilation
# sc_launch_test call
################################################
nbtoy=0
for toy in ${USER_TOY[@]}; do
          echo "++++++++++++++++++++++++++++++++++++++++"
```

```
        echo "++++++++++++++++++++++++++++++++++++++"
        echo "toy :" $toy
        export OASIS_TOYDIR=${toy}
        export casename=`basename $toy`
        echo "casename :" $casename
        echo "++++++++++++++++++++++++++++++++++++++"
        echo "++++++++++++++++++++++++++++++++++++++"
        export pathname=`dirname $toy`
        export paramloc=${USER_PARAMLOC[$nbtoy]}
        echo "Localisation of the parameter test files for the toy" $paramloc
        export nbttot=${USER_TEST[$nbtoy]}
        echo "Total number of tests " $nbttot " for toy " ${casename}
        for nb_tests in $( eval echo {1..${nbttot}} ); do
                echo "Test number : " ${nb_tests}
            cd $paramloc
                if [ ${nb_tests} -le 9 ]; then
                        . ./param_${casename}_test0${nb_tests}
                else
                        . ./param_${casename}_test${nb_tests}
                fi
         # Compilation of the toy for this test
                echo "OASIS_TOYDIR : ${OASIS_TOYDIR}"
                echo "USER_MAKELOC : ${USER_MAKELOC}"
            cd ${OASIS_TOYDIR}
         echo "Compile $casename on ${OASIS_ENV}"
         if [ ${nb_tests} -le 9 ]; then
              ./oasis_test_build param_${casename}_test0${nb_tests}
         else
              ./oasis_test_build param_${casename}_test${nb_tests}
         fi
################################################
                if [ ${nb_tests} -le 9 ]; then
                        ./oasis_test_run.sh param_${casename}_test0${nb_tests}
                else
                        ./oasis_test_run.sh param_${casename}_test${nb_tests}
                fi

    done
        (( nbtoy=nbtoy+1 ))
done
################################################
################################################
```

## 7.8 Content of sc_compile_oasis.sh

```
################################################
# Compilation of OASIS or PYOASIS
################################################
cd ${OASIS_COUPLE}/util/make_dir
make realclean -f ${OASIS_COUPLE}/util/make_dir/TopMakefileOasis3
make -f ${OASIS_COUPLE}/util/make_dir/TopMakefileOasis3 $OASIS_TARGET
```

```
if [ -z "${OASIS_TARGET}" ]; then
# results in INSTALL_OASIS.${OASIS_ENV}: build-static  include  lib
#
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/libmct.a
if [ `echo $?` -ne 0 ]; then
   echo "pb libmct.a not created"
   exit 1
fi
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/libmpeu.a
res_command=`echo $?`
if [ ${res_command} -ne 0 ]; then
   echo "pb libmpeu.a  not created"
   exit 1
fi
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/libscrip.a
res_command=`echo $?`
if [ ${res_command} -ne 0 ]; then
   echo "libpsmile.MPI1.a not created"
   exit 1
fi
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/libpsmile.MPI1.a
res_command=`echo $?`
if [ ${res_command} -ne 0 ]; then
   echo "libscrip.a not created"
   exit 1
fi
else
# results in INSTALL_OASIS.${OASIS_ENV} : build-shared  include  lib  python
#
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/libmct.so
if [ `echo $?` -ne 0 ]; then
   echo "pb libmct.a not created"
   exit 1
fi
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/libmpeu.so
res_command=`echo $?`
if [ ${res_command} -ne 0 ]; then
   echo "pb libmpeu.a  not created"
   exit 1
fi
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/libscrip.so
res_command=`echo $?`
if [ ${res_command} -ne 0 ]; then
   echo "libpsmile.MPI1.a not created"
   exit 1
fi
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/libpsmile.MPI1.so
res_command=`echo $?`
if [ ${res_command} -ne 0 ]; then
   echo "libscrip.a not created"
   exit 1
```

```
fi
ls ${OASIS_COUPLE}/INSTALL_OASIS.${OASIS_ENV}/lib/liboasis.cbind.so
res_command=`echo $?`
if [ ${res_command} -ne 0 ]; then
   echo "libscrip.a not created"
   exit 1
fi
#
# End else test on pyoasis
fi
```

## 7.9 Content of env_tests_param_tioman_intel21.1.1_intelmpi2021.1.1

```
################################################
## USER SECTION
################################################
# OASIS VARIABLES DEFINITION
# + NAMES OF THE DIFFERENT TOYS
# We need at least grids.nc for each toy to calculate the analytical function
################################################
# OASIS_ROOT
export OASIS_ROOT=/space/coquart
# Repository with OASIS3-MCT sources
export OASIS_COUPLE=${OASIS_ROOT}/oasis3-mct
# OASIS_ENV used to launch the environment to test on different machines
# (see sc_launch_tests.sh)
export OASIS_ENV=tioman_intel21.1.1_intelmpi2021.1.1
# Repository with OASIS3-MCT toys
export OASIS_TESTS=/space/coquart
# To compile OASIS or not
export OASIS_COMPILE=TRUE
# Variables to defined the compilation options of OASIS3-MCT and the toys
# TRUE or nothing
export OASIS_DEBUG=TRUE
export OPENMP=TRUE
# Variable to wait the end of the test before submitting the next one
export OASIS_SUBMITWAIT=1
# Localization and name of the toys
export USER_TOY=("${OASIS_TESTS}/oasis3-mct_tests/toy_interpolation")
# Number of tests for each toy
export USER_TEST=("104")
# Localization of the file paramater for each toy
export USER_PARAMLOC=("${OASIS_TESTS}/oasis3-mct_tests/toy_interpolation")
# To run the tests in USER_RUNDIR
export USER_RUNDIR=${OASIS_ROOT}/OA3_MCT_RES
```