

ESMF Cheat Sheet

by Charlie Zender
University of California at Irvine

Department of Earth System Science
University of California
Irvine, CA 92697-3100

zender@uci.edu
Voice: (949) 824-2987
Fax: (949) 824-3256

Contents

1	Introduction	1
2	Environment	2
2.1	Bash	2
3	Models	3
3.1	CCSM Component Models	3
3.1.1	CAM: Community Atmosphere Model	4
3.1.2	CLM: Community Land Model	4
3.2	CCSM: Community Climate System Model	4
3.2.1	ESMF-specific CCSM modifications	5
3.2.2	Creating a new CCSM case	6
3.2.3	Resolved Scripts	7
3.2.4	Run Geometry	7
3.2.5	Testing CCSM	7
3.2.6	Submitting Run Scripts	7
3.2.7	Archiving Data	8

List of Tables

1 Introduction

The [Earth System Modeling Facility](#) (ESMF) is designed to run large climate simulation codes. This document describes some general usage pointers for the ESMF, then demonstrates how to install and run certain codes (Section [3](#)).

2 Environment

Users must have and maintain standard shell environment files. Essentially these files provide path information and helpful shortcuts to the users and any batch scripts that invoke them. These files need not be overly complex. Most importantly, they must set the `${PATH}` environment variable and pass control on to CCSM scripts without errors.

Shell environment files must work in batch (i.e., non-interactive) mode. In particular, they must not cause any output to `stdout` in batch mode—or the batch environment will be incorrect and cause unpredictable results. Minimal shell environment files for the ESMF are located in your home directory. Following are some suggested additions.

2.1 Bash

The default login shell on the ESMF is Bash. Add these lines to your default standard shell file `.bashrc`, or `.profile`:

```
alias bsrc="source ${HOME}/.bashrc"
alias cp='cp -i' # -i = inquire
alias cvc='cvs commit -m ""'
alias cvu='cvs update -kk'
alias env='env | sort | more'
alias h='history'
alias ls='ls -F' # -F = mark directories with /, links with @, etc.
alias m='less'
alias make='gmake'
alias mv='mv -i' # -i = inquire
alias rm='rm -i' # -i = inquire
alias scp='scp -C' # -p = preserve mode, time, -C = enable compression
alias tar='gtar'
export CVS_RSH='ssh'
export CVSUMASK='002' # Default file permissions for CVS
export HOST='/bin/hostname'
export NETCDF_INC='/usr/local/include'
export NETCDF_LIB='/usr/local/lib'
export PS1='\u@\h:\w\$ ' # Prompt user@host:cwd$ NeR98 p. 71 NB: single quotes
```

Many ESMF users will want to run models, such as CCSM, which use C Shell scripts. For these scripts to work in batch mode, users must have a valid `.cshrc` file as well.

```
alias csrc "source ${HOME}/.cshrc"
alias cp 'cp -i' # -i = inquire
alias cvc 'cvs commit -m ""'
alias cvu 'cvs update -kk'
alias env 'env | sort | more'
alias h 'history'
alias ls 'ls -F' # -F = mark directories with /, links with @, etc.
```

```
alias m 'less'
alias make 'gmake'
alias mv 'mv -i' # -i = inquire
alias rm 'rm -i' # -i = inquire
alias scp 'scp -p -C' # -p = preserve mode, time, -C = enable compression
alias tar 'gtar'
setenv CVS_RSH 'ssh'
setenv CVSUMASK '002'
setenv HOST '/bin/hostname'
setenv NETCDF_INC '/usr/local/include'
setenv NETCDF_LIB '/usr/local/lib'
set prompt=${USER}@${HOST}:${cwd}\$ " "
```

Users with a `.login` file in their home directory should ensure that it executes error free.

Alternatively, users can change their login shell to `cs` or `tcsh` vis the standard ESMF procedure

```
ssh esmfews
passwd -s
```

3 Models

This section describes how to install and run some of the various climate, chemical, and biogeochemical models for which the ESMF is designed.

3.1 CCSM Component Models

This section documents how to install and run the component models which compose the fully coupled NCAR CCSM (Community Climate System Model). Running the fully coupled CCSM itself is described in Section 3.2. First we describe the preliminary system modifications common to all CCSM component models.

CCSM component models may run in Symmetric Multi-Processor (SMP), Single Program Multiple Data (SPMD), and hybrid (i.e., combined SMP and SPMD) modes. To take advantage of the SPMD capabilities, make sure you have the Message Passing Interface (MPI) installed. MPI is already installed on the ESMF. To install it on a Debian GNU/Linux machine use:

```
apt-get install mpich # Debian GNU/Linux
```

CCSM build scripts may require the GNU Make executable to be named `gmake` rather than `make`. `gmake` is installed in `/usr/local/bin` on the ESMF. To point the GNU `make` executable on a Debian GNU/Linux machine to `gmake` use, e.g.,

```
sudo ln -s /usr/bin/make /usr/bin/gmake
```

All CCSM input-output (I/O) is performed with the `netCDF` data model. `netCDF` is already installed on the ESMF. To install `netCDF` on a Debian GNU/Linux machine use:

```
apt-get install netcdf # Debian GNU/Linux
```

3.1.1 CAM: Community Atmosphere Model

No documentation yet. Please contribute!

3.1.2 CLM: Community Land Model

The Community Land Model (CLM) is described by *Dai et al.* [2003]. Other important references are *Bonan* [2002], *Bonan* [1996], and *Bonan* [1998].

The most frequently used WWW sites pertinent to CLM are

1. [ESMF Homepage](#)
2. [CLM Homepage](#)
3. [LSM \(CLM's predecessor\) Homepage](#)
4. [CLM 2.1 Source Code](#)
5. [CLM 2.1 User's Guide \(CLMUG\)](#)
6. [NCL CCSM Graphics](#)
7. [NCO netCDF Operators](#)

Make ESMF account capable of running CLM offline:

```
# Download and unpack CLM 2.1 source code into your home directory
cd ~
http://www.cgd.ucar.edu:8080/accept/license?action=fillOut&file_id=7
gtar xvzf CLM2.1_code.tar.gz

# Create model run space
if [ -n "${LOGNAME}" ]; then export LOGNAME=${USER}; fi
mkdir /ptmp/${LOGNAME}

# Create local script directory
mkdir ~/clm; cd ~/clm;
cp ~/zender/ess_lsp/clm.sh .
cp ~/zender/ess_lsp/clm.txt .

# Modify default CLM Makefile with ESMF paths
cp ~/zender/ess_lsp/clm_Makefile.mk ${HOME}/clm2/bld/offline/Makefile

# Build and run model
cd ~/clm; ./clm.sh
```

3.2 CCSM: Community Climate System Model

In modifying the CCSM to run on the ESMF, it is helpful to know that the ESMF and the NCAR-supported machine *bluesky* have nearly identical hardware and software configurations. First, *esmf* and *bluesky* are both clusters of p655 (8-way) and p690 (32-way) nodes connected by Federation switches. Each 8-way *esmf* node in the compute queues has 16 GB

RAM (the interactive 8-way node `esmf04m`, which is on the internet as `esmf.ess.uci.edu`, has twice as much RAM, 32 GB). The 32-way `p690` node has 64 GB RAM. Hence, each computational CPU on the ESMF has 2 GB RAM available. `bluesky` has about twenty times as many nodes as `esmf`. Finally, `esmf` has faster CPUs than `bluesky` (1.5 GHz rather than 1.3 GHz).

3.2.1 ESMF-specific CCSM modifications

With this knowledge, we created machine files which describe the `esmf` cluster and queueing architecture to the CCSM. These files are based on the corresponding `bluesky` machine files distributed with the CCSM. Five machine-specific files are required to make the `esmf` a valid CCSM machine-type. These files are

1. `Macros.AIX` (`ccsm3_0/models/bld/Macros.AIX`)
2. `check_machine` (`ccsm3_0/scripts/ccsm_utils/Tools/check_machine`)
3. `batch.ibm.esmf` (`ccsm3_0/scripts/ccsm_utils/Machines/batch.ibm.esmf`)
4. `env.ibm.esmf` (`ccsm3_0/scripts/ccsm_utils/Machines/env.ibm.esmf`)
5. `run.ibm.esmf` (`ccsm3_0/scripts/ccsm_utils/Machines/run.ibm.esmf`)

First, read the purpose of the ESMF customizations. This will be helpful to those attempting to port the CCSM to other machines (e.g., Linux clusters) at UCI. Then, copy the ESMF-customized files to your local CCSM source tree.

The file `Macros.AIX` controls the options the compiler uses to build the CCSM source code (both C and Fortran). The first change replaces the default NCAR `netCDF` library path, `/usr/local/lib64/r4i4`, with its ESMF equivalent, `/usr/local/lib`:

```
mv ~/ccsm3_0/models/bld/Macros.AIX ~/ccsm3_0/models/bld/Macros.AIX.orig
cp ~/zender/ccsm3_0/models/bld/Macros.AIX ~/ccsm3_0/models/bld/Macros.AIX
```

All ESMF system libraries use the 64-bit ABI where possible. Hence NCAR's `lib64` is simply the ESMF's `lib`. Second, the default size for Fortran variables of type `real` is four bytes, i.e., single precision. The default size for Fortran variables of type `integer` is also four bytes. Newer scientific computing codes, such as CCSM, explicitly specify the sizes of integers and reals so the default sizes are never used. However, many old codes rely on compiler-defaults to specify the sizes of `integer` and `real`. Some applications expect other defaults, e.g., eight-byte `real` and four-byte integer or `r8i4` in NCAR parlance. The ESMF uses the `r4i4` standard for all libraries in `/usr/local`. This obsoletes the need to explicitly specify an equivalent to NCAR's `r4i4` subdirectory.

The file `check_machine` is used by the `create_newcase` script to customize the script. If `esmf` is not included in `check_machine`, then `create_newcase` will exit with the error `check_machine ERROR, esmf not supported`.

The file `batch.ibm.esmf` contains most of the default header that will be used by the CCSM run script when the script is run in batch mode. Here is where the default `LoadLeveler` commands are set. The ESMF uses the default CCSM settings for the IBM AIX machine `bluesky` except for the following changes:

1. Default class (i.e., queue) is `com_rg8` rather than `cs1_rg${max_tasks_per_node}`

2. Default `wall_clock_limit` is 21600 s (6 hr), rather than 7200 s.
3. Job accounting is disabled by commenting out the line containing `jareport`.

The file `env.ibm.esmf` contains environment variables which determine how the CCSM is built and where it looks for input data and stores output data. `env.ibm.esmf` is the same as `env.ibm.bluesky` with the following exceptions

1. `GMAKE_J` determines how many jobs (in effect, processors) the parallelization of the compilation process will request. This is set to eight but should be set to four if you are often compiling interactively (rather than in the compute queues).
2. `DIN_LOC_ROOT` is `/datashare/inputdata/csm` instead of `/fs/cgd/csm/inputdata`
3. `DIN_LOC_ROOT_USER` is `/datashare/inputdata_user` instead of `/fs/cgd/csm/inputdata_user`
4. `DIN_LOC_MSROOT` is `/datashare/inputdata/csm` instead of `/CCSM/inputdata`

Finally, the CCSM versions of `env.ibm.*` contain a subtle bug in the task geometry clauses. Essentially, C Shell syntax splits the two `set` statements that look like the body of many `if-endif` clauses into two. The first statement is inside the condition and the second is outside and thus always executed. This bug causes geometry statements to overwrite each other. We re-coded `env.ibm.esmf` to avoid this confusion.

The file `run.ibm.esmf` is the template run script. The fully resolved run script will be created from this template. This template handles all the details of building, executing the model, as well as storing the output data and possibly continually re-submitting the job until completion. `run.ibm.esmf` is identical to `run.ibm.bluesky`.

Copy these three files to your local source tree:

```
for fl in batch.ibm.esmf env.ibm.esmf run.ibm.esmf ; do
  mv ~zender/ccsm3_0/scripts/ccsm_utils/Machines/${fl} \
      ~/ccsm3_0/scripts/ccsm_utils/Machines/${fl}.orig
  cp -f ~zender/ccsm3_0/scripts/ccsm_utils/Machines/${fl} \
      ~/ccsm3_0/scripts/ccsm_utils/Machines
done
```

Normally, no further customization is necessary. These `esmf`-customized defaults are sufficient to get started running the CCSM. However, advanced CCSM users will often need to further modify these `esmf`-customized defaults.

3.2.2 Creating a new CCSM case

We are now ready to create the new CCSM case, i.e., experiment. First, store the case name the `$CASEID` environment variable. This name should be clueful about the experiments' purpose. Next, use the `create_newcase` script which will create environment variable files and generate the `configure` script for this experiment.

```

cd ~/ccsm3_0/scripts
export CASEID='T31x3'
export CASEID='T42x1_40'
export CASEID='T85x1_64'
export CASEROOT="/ptmp/${USER}/${CASEID}"
./create_newcase -case ${CASEROOT} -mach esmf -res T31_gx3v5 -compset B
./create_newcase -case ${CASEROOT} -mach esmf -res T42_gx1v3 -compset B
./create_newcase -case ${CASEROOT} -mach esmf -res T85_gx1v3 -compset B

```

This will create the `$CASEROOT` directory. Before the model may be run, however, it is necessary to populate the `$CASEROOT` directory with the resolved scripts.

3.2.3 Resolved Scripts

Each `$CASEID` may have multiple resolved-scripts generated for it. The resolved scripts are customized for particular initialization sequences (e.g., restart runs), resolution, machine, and component sets. The `configure` command generates the resolved scripts:

```

cd ${CASEROOT}
./configure -cleanmach esmf # Cleanup any old builds for esmf
./configure -mach esmf

```

3.2.4 Run Geometry

3.2.5 Testing CCSM

CCSM supports a sophisticated test suite.

```

./create_test -testname TER.01a.T31_gx3v5.B.esmf -testroot /ptmp/zender/tstinstall
./create_test -testname TDB.01a.T31_gx3v5.B.esmf -testroot /ptmp/zender/tstinstall
./create_test -testname TBR.01a.T31_gx3v5.B.esmf -testroot /ptmp/zender/tstinstall
./create_test -testname TBR.02a.T31_gx3v5.B.esmf -testroot /ptmp/zender/tstinstall
./create_test -testname THY.01a.T31_gx3v5.B.esmf -testroot /ptmp/zender/tstinstall
./create_test -testname THY.02a.T31_gx3v5.B.esmf -testroot /ptmp/zender/tstinstall

```

Passing a single test is a good indication that your CCSM environment and installation are correct. Passing all tests means

3.2.6 Submitting Run Scripts

Submit the job to the ESMF LoadLeveler queues with `llsubmit`:

```

cd ${CASEROOT}
llsubmit T31x3x.esmf.run

```

Check the status of the job in the queue with `llq`.

All CCSM component models create log files during execution. Examine these log files to ensure your job is proceeding smoothly. The coupler log is most informative about the overall progress of the simulation.

```
cd ${CASEROOT}/cpl
more cpl.log.041229-190833
```

Log files are stamped with the job-submission time. The time-stamp has the format YYMMDD-HHMMSS. Hence each simulation has a unique identifier, such as 041229-190833 above. The time-stamp is consistent among all the model components in the same simulation. After the simulation completes, the model compresses the log files and stores them, along with the history tapes (i.e., output data) in /ptmp/\${USER}/archive/\${CASEID}.

3.2.7 Archiving Data

CCSM, by default, stores short term output files in

```
/ptmp/${USER}/archive/${CASEID}
```

CCSM also has automated long term archiving capability. This can be customized to copy data and logs to any computer on the Internet. Once the ESMF disk farm is working, we will activate automatic long-term archiving to

```
/data/${USER}/archive/${CASEID}
```

References

- Bonan, G., *Ecological Climatology: Concepts and applications*, 678 pp., Cambridge University Press, Cambridge, UK, 2002.
- Bonan, G. B., A land surface model (LSM version 1.0) for ecological, hydrological, and atmospheric studies: Technical description and user's guide, *Tech. Rep. NCAR/TN-417+STR*, National Center for Atmospheric Research, Boulder, Colo., 1996.
- Bonan, G. B., The land surface climatology of the NCAR Land Surface Model coupled to the NCAR Community Climate Model, *J. Clim.*, 11(6), 1307-1326, 1998.
- Dai, Y., et al., The Common Land Model (CLM), *Submitted to Bull. Am. Meteorol. Soc.*, 84(8), 1013-1023, 2003.

Index

configure, 7
create_newcase, 6
csh, 3
gmake, 3
jareport, 6
llq, 7
llsubmit, 7
make, 3
stdout, 2
tcs, 3
.bashrc, 2
.cshrc, 2
.login, 3
.profile, 2
/CCSM/inputdata, 6
/datashare/inputdata/csm, 6
/datashare/inputdata_user, 6
/fs/cgd/csm/inputdata_user, 6
/fs/cgd/csm/inputdata, 6
/usr/local/bin, 3
/usr/local, 5
Macros.AIX, 5
batch.ibm.esmf, 5
check_machine, 5
configure, 6
create_newcase, 5
env.ibm.bluesky, 6
env.ibm.esmf, 6
run.ibm.bluesky, 6
run.ibm.esmf, 6
bluesky, 4, 5
esmf04m, 5

Bash, 2

C Shell, 2
\$CASEID, 6
\$CASEROOT, 7
CCSM, 3
class, 5
CLM, 4
Community Climate System Model, 3
Community Land Model, 4
coupler, 7
Debian, 3
ESMF, 1
GNU/Linux, 3
hybrid, 3
I/O, 3
LoadLeveler, 5, 7
Make, 3
Message Passing Interface, 3
MPI, 3
queue, 5
resolved scripts, 7
run script, 5, 6
Single Program Multiple Data, 3
SMP, 3
SPMD, 3
Symmetric Multi-Processor, 3
wall_clock_limit, 6