

# Art of Climate Modeling 2006

## CCSM3 Tutorial: How to Build and Run the Model

Christine Shields NCAR/CGD/CCR

shields@ucar.edu

# Webpage Documentation

CCSM Homepage:

<http://www.cgd.ucar.edu/csm>

CCSM3 Public Release Homepage:

<http://www.cesm.ucar.edu/models/ccsm3.0>

CCSM3.0 Documentation Homepage:

<http://www.cesm.ucar.edu/models/ccsm3.0/ccsm>

Note: The documentation page contains the User's Guide in html, pdf, or ps formats.

# Overview

- A. Review CCSM3 Build Exercise
- B. CCSM3 Runtime Overview
- C. How to Run a “Startup” Case
- D. How to Run a “Hybrid” Case
- E. How to view your output
- F. How to process your output
- G. Exercises

# A. Review Build: Simple Learning Case

Fully coupled CCSM3 Low Resolution Model

Case name = b30.btst

Resolution: T31\_gx3v5

Machine: Bluesky (NCAR IBM supercomputer)

A fully coupled model will engage all active component models:

cam: atmosphere

clm : land

csim: ice

pop: ocean

cpl: coupler

The coupler allows the other four component models to communicate with each other.

## A.1. Where to Find Model Code

- 1) Log on to roy, then bluesky using your crypto-card

(Note: if you do not know how to do this, please go to the following website: [http://atoc.colorado.edu/~dcn/ACM/wiki/index.php/Main\\_Page](http://atoc.colorado.edu/~dcn/ACM/wiki/index.php/Main_Page))

- 2) `cd` (unix command, change directory) to  
`/fis/cgd/cseg/csm/collections/ccsm3_0_1_beta19`
- 3) You are now in the source code tree for the version of the model you will be using. Read the README file.

## A.2. Where to Find the Initial Datasets

- 1) You should already be logged onto bluesky
- 2) `cd to /fis/cgd/cseg/csm/inputdata`
- 3) Feel free to explore these directories. Initial datasets for component models are kept in their respective directories.
- 4) You will NOT need to do anything to these files and should NOT attempt to write into these directories. Writing permission into these directories are for ccsn personnel only.

## A.3. CCSM Case Directory Organization: Overview

There are 3 sets of directory trees you will need to understand to build the model.

- 1) Where your build/case scripts reside
- 2) Where the model code/input resides
- 3) Where the executables reside

## A.4. CCSM Case Directory Organization: Scripts

For the purposes of this course, your case and build scripts will reside in your home directory tree. This will be the “\$CASEROOT” directory. “\$CASEROOT” is one of many environmental variables used by the model (denoted by the “\$” in front of the variable name). The model build procedure will automatically set these for you if you follow the steps in this tutorial.

1. **Type/enter “cd”; then, type/enter “pwd”.** “pwd” will show your current working directory. It should be “/home/bluesky/<logname>” where logname is your bluesky account login name. For example, if your login name is “smith”, then you should be in the directory “/home/bluesky/smith” .
2. **Type/enter “mkdir -p ccsm3/scripts”.** You have now just created the parent directory for your ccsm3 simple case. This will be where your “\$CASEROOT” resides. We will get back to this directory after we have reviewed the other directory structures for CCSM3.

## A.5. CCSM Directory Organization: Model

- You should already know where the model source code resides from B.1. You should also understand the model directory structure from the “README” file found in this directory. This is your “\$CCSMROOT” directory.
- You will NOT be editing/modifying anything in this directory.
- You WILL be USING the tool called “create\_newcase” in the “**\$CCSMROOT/scripts**” subdirectory.
- **Note: \$CCSMROOT/scripts = /fis/cgd/cseg/csm/collections/ccsm3\_0\_1\_beta19/scripts.**
- The create\_newcase tool, when used properly, essentially creates your \$CASEROOT directory, i.e. **/home/bluesky/<logname>/ccsm3/scripts/\$CASE**. In addition to creating the \$CASE directory (where \$CASE is the case name for this exercise), create\_newcase deposits the necessary files and tools into \$CASEROOT which will consequently be used to create your build and case scripts.
- Details on using this tool will be covered in section D of this document.

## A.6. CCSM Directory Organization: Executables

- After you have successfully created the build and case scripts in your \$CASEROOT directory, you will be able to build the model.
- The model builds (and runs) in a temporary directory on bluesky called “ptmp”.
- You should have a temporary directory on “ptmp”. To check, **type/enter “cd /ptmp/<logname>”** where logname is your login name.
- Although the build script resides in your \$CASEROOT directory, the build procedure operates in the ptmp space . All necessary ptmp subdirectories will be created for you by the build procedure.

## A.7. How to create your simple case: Using create\_newcase

- 1) Go to the model source code directory, (i.e. this will be eventually defined as \$CCSMROOT)  
type/enter “**cd /fis/cgd/cseg/csm/collections/ccsm3\_0\_1\_beta19**”
- 2) Go to the scripts subdirectory where the utility create\_newcase resides.  
type/enter “**cd scripts**”
- 3) Look at the help information for this utility,  
type/enter “**create\_newcase -help**”
- 4) For the purposes of our course, we will have you create a fully coupled, low resolution test case on bluesky. You will now begin to create your build/case scripts in your home directory tree, in what will be defined as your \$CASEROOT directory.

```
type/enter (all one line) “create_newcase -case  
/home/bluesky/<logname>/ccsm3/scripts/b30.btst -mach bluesky -res T31_gx3v5  
-compset B -ccsmroot /fis/cgd/cseg/csm/collections/ccsm3_0_1_beta19”
```

where <logname> = your login name  
b30.btst = case name for this exercise

The nomenclature, “b” signifies component set B (fully coupled), “30” for version 3.0 of ccsm, “btst”, short descriptor for build testcase, which is the purpose of this exercise.

## A.8. How to create your simple case: Looking at what create\_newcase did...

- 1) Type/enter “**cd**  
**/home/bluesky/<logname>/ccsm3/scripts/b30.btst**”
- 2) Type/enter “**ls -al**”. You should see the following files and directories:
  - .cache (directory)
  - SourceMods (directory)
  - configure (tool)
  - env.readme (documentation on enviromental variables)
  - env\_conf (configure enviromental variables)
  - env\_mach.bluesky (machine enviromental variables)
  - env\_run (runtime enviromental variables)
- 3) During our class, we will learn more detail about each of these files, but for this exercise, we will simply use the default settings.

## A.9. How to create your simple test case: configure

We will now configure our simple learning case using the tool “configure”.

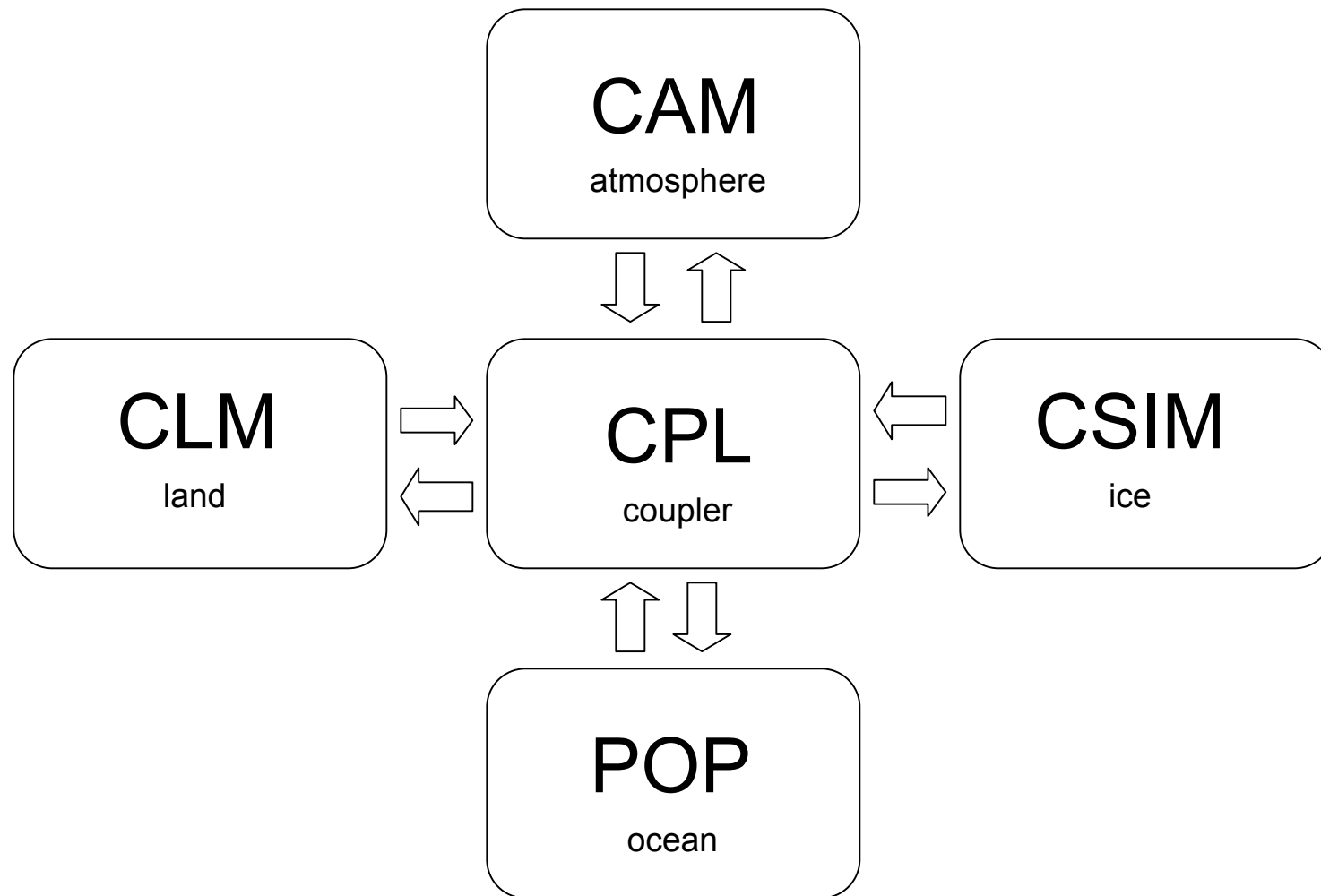
- 1) We will stay in our \$CASEROOT directory,  
“/home/bluesky/<logname>/ccsm3/scripts/b30.btst” .
- 2) Type/enter, “**configure –help**” to review the help documentation
- 3) Type/enter, “**configure –mach bluesky**” to configure your case
- 4) After you run this command, you will see several new directories and files. You now should have all you need to build the model. Type “**ls –al**”, to review the contents of your \$CASEROOT.  
New directories include: Buildexe, Buildlib, Buildnml\_Prestage  
New files include: b30.btst.bluesky.build, b30.btst.bluesky.l\_archive,  
b30.btst.bluesky.run.
- 5) Feel free to explore your \$CASEROOT and all its files and subdirectories. In class, we will discuss the contents of \$CASEROOT in more detail, but for the purpose of this exercise, we will only look at the build script, **b30.btst.bluesky.build**.

## A.10. How to build your simple case executables

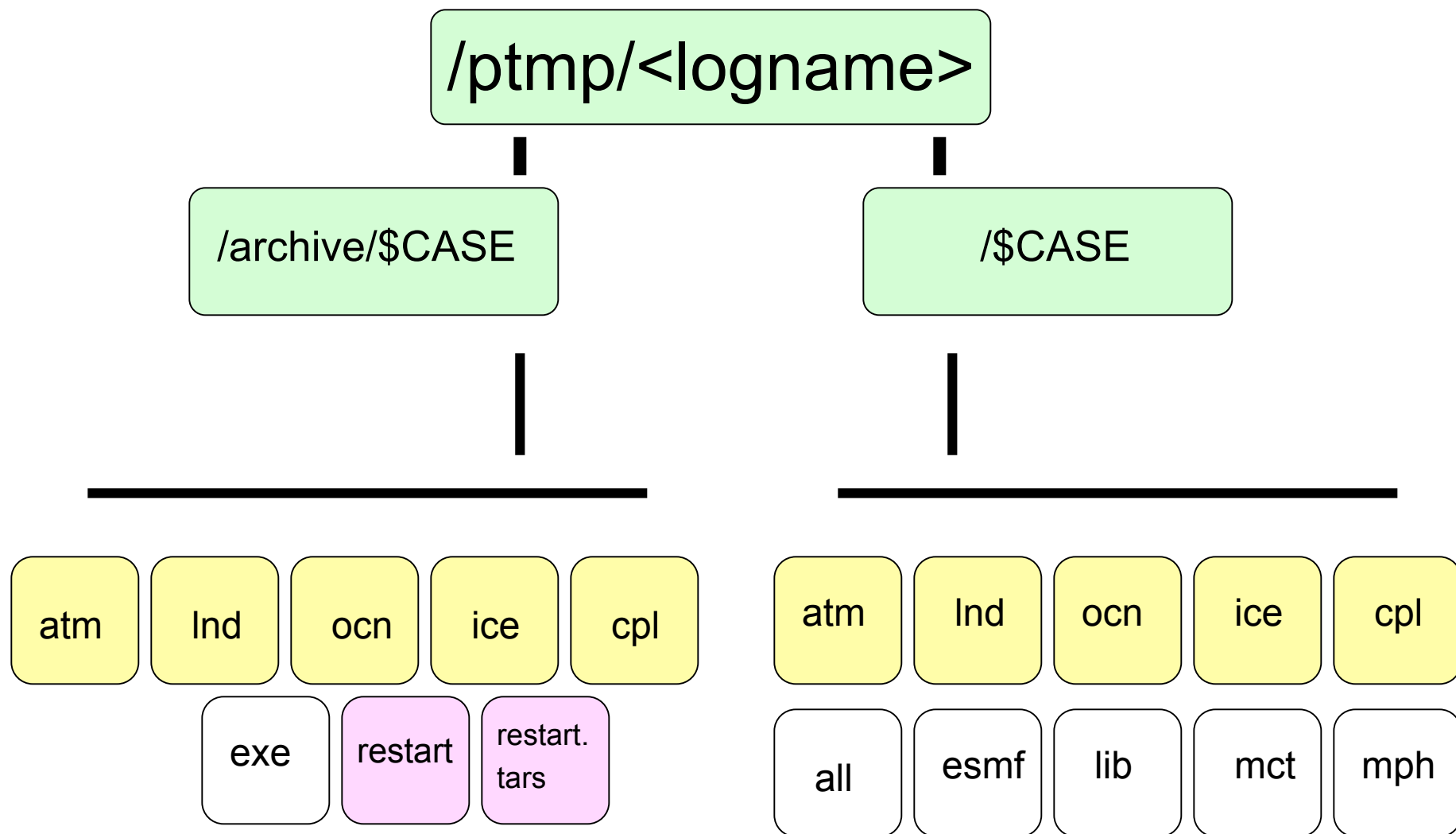
This part is easy, we have done all the work with `create_newcase` and `configure...`

- 1) We will stay in `$CASEROOT`
- 2) Optional: open a new (second) bluesky window (see comment 5).
- 3) Type/enter “**b30.btst.bluesky.build**”... the model should now be building all the executables in the `ptmp` directory.
- 4) You will see a steady stream of `stdout` to your screen from the `b30.btst.bluesky.build` command. The entire build process should take up to 15 minutes.
- 5) To check your `ptmp` directory while the model is building, and follow the progress noted by the `stdout`, open a new bluesky window, and “**cd /ptmp/<logname>/b30.btst**”. There will be several layers of subdirectories produced by the build procedure. Feel free to explore your `ptmp` directories.

## B. CCSM3 Runtime Overview



## B.1. CCSM Overview: Short-Term Archive and Runtime Directories



## B.2.1 CCSM OVERVIEW: Archiving

Look in your `env_mach.bluesky` file...

`$DOUT_S_ROOT`

`$DOUT_L_MSROOT`  
`$DOUT_L_RCP_ROOT`

`$DOUT_S_ROOT = /ptmp/<logname>/archive/$CASE`

`$DOUT_L_MSROOT = /<LOGNAME>/csm/$CASE`

Example, if your logname is smith, and `$CASE = b30.btst`, then

`$DOUT_L_MSROOT = /SMITH/csm/b30.btst`

## B.2.2. CCSM3 Overview: Archiving

- 1) Upon model execution, the model will create your /ptmp archive directories for you automatically.
- 2) By default, after model job completion, the model will copy your output files into your short term archiving directories (\$DOUT\_S\_ROOT).
- 3) The model will **NOT** automatically write your files to the long-term archive. To have the model automatically invoke the script which archives to long-term storage, edit the **env\_mach.bluesky** parameter, **\$DOUT\_L\_MS** or **\$DOUT\_L\_RCP** to “TRUE”.
- 4) If you are running at NCAR, the long-term archive is called the Mass Storage System (MSS). In the case, you would set \$DOUT\_L\_MS to “TRUE”.
- 5) The long term archiving script will always leave a *copy* of the *latest* set of model output in the short term archive directory.
- 6) The long-term archiving script is in your \$CASEROOT directory and is called b30.btst.bluesky.l\_archive. Remember, as mentioned previously, the long term archiving script is submitted automatically by the model run scripts. We will discuss this more in the next section.

## B.3. CCSM3 Overview: Run type

### Startup

Initial Run, everything starts from initial conditions.

### Continue

The continuation of a previous run... a continue run starts from “restart” files.

### Branch

A model case where a new run is started but uses restart files from another case. This is done to enable the model to start from a climate state generated from a previous model run, but keeps the timestamp from the restart files.

### Hybrid

A combination of Startup and Branch. The atm/Ind components start from initial conditions. The ocn/ice components start from restart files. Again, this is done to enable the model to start from a climate state generated from a previous model run, however hybrid runs allow for the model to use any \$RUN\_STARTDATE.

## C. How to Run a “Startup” Case

We will run a startup case for one month followed by two one-month restart cases.

The first restart case we submit by hand, the second restart case we will be done using an automatic resubmit feature.

# C.1. How to Run a “Startup” Case: Scripts

1) Run `create_newcase` to get basic scripts:

a. our case name will be `b30.strst`

b. our `$CASEROOT =`

`/home/bluesky/<logname>/ccsm3/scripts/b30.strst`

c. our `$CCSMROOT =`

`/fis/cgd/cseg/csm/collections/ccsm3_0_1_beta19`

How do we do this?

## C.1.2. How to Run a “Startup” Case: Scripts

We do the following:

cd to our ccsroot scripts directory...

```
cd /fis/cgd/cseg/csm/collections/ccsm3_0_1_beta19/scripts
```

```
create_newcase -case  
/home/bluesky/<logname>/ccsm3/scripts/b30.strst -mach  
bluesky -res T31_gx3v5 -compset B -ccsmroot  
/fis/cgd/cseg/csm/collections/ccsm3_0_1_beta19
```

## C.1.3. How to Run a “Startup” Case: Scripts

2) Let’s look at the `env_conf` file.... (cd back to our `$CASEROOT` dir)

`casename`, model component names (active or “data” models), `grid`, and runtime variables are specified.

`create_newcase` specifies `casename`, `grid`, and component set, so these are automatically set for you.

`create_newcase` does NOT specify `run_type`, therefore, you need to specify this by editing the **`env_conf`** file. For our current example, “startup”, is the default setting, so we don’t need to modify **`env_conf`** for this exercise. However, if you were to run a “branch” or “hybrid” case, you would modify this file.

NOTE: “Continue” runs are NOT specified in the file. Continue runs are simply continuations of startup, hybrid, or branch runs. They are not separate (unique) runs and are considered the same case as its initial run (i.e., startup, hybrid, or branch). The only thing different about a “continue” run is the fact that you start from restart files, not initial files. “Continue” runs are controlled by **`env_run`** ...

Why do we use “continue runs”?

# env\_conf

```
#!/bin/csh -f

# Documentation of following environment variables is provided in env.readme

setenv CASE b30.strst
setenv CASESTR b30.strst
setenv COMP_ATM cam
setenv COMP_LND clm
setenv COMP_ICE csim
setenv COMP_OCN pop
setenv COMP_CPL cpl
setenv CSIM_MODE prognostic
setenv GRID T31_gx3v5
setenv RUN_TYPE startup # [startup,hybrid,branch]
setenv RUN_STARTDATE 0001-01-01 # [yyyy-mm-dd]
setenv RUN_REFCASE case.std # [only applies to hybrid or branch runs]
setenv RUN_REFDATE 0001-01-01 # [only applies to hybrid or branch runs]
setenv IPCC_MODE OFF # [OFF, 1870_CONTROL, 1870_TO_PRESENT, RAMP_CO2_ONLY,
# FUTURE_A1, FUTURE_A2, FUTURE_B1, FUTURE_B2]
setenv RAMP_CO2_START_YMD 00000000 # Start date of CAM CO2 ramp in form YYYYMMDD
# This variable MUST be set if IPCC_MODE is RAMP_CO2_ONLY
# This variable is ignored if IPCC_MODE is not RAMP_CO2_ONLY

#=====
# DERIVED ENVIRONMENT VARIABLES (user should not edit these)
#=====

setenv ATM_GRID `echo $GRID | sed s/.\`*_//`; setenv LND_GRID $ATM_GRID
setenv OCN_GRID `echo $GRID | sed s/.\`*_//`; setenv ICE_GRID $OCN_GRID
```

## C.1.4. How to Run a “Startup” Case: Scripts

### 3) Edit env\_run

1. change SETBLD to **AUTO**
2. change STOP\_OPTION to **nmonths**
3. change STOP\_N to **1**
4. change HIST\_OPTION **monthly**
5. change AVHIST\_OPTION **monthly**

Note 4) and 5) affect the COUPLER only, history output for all other components are controlled by the component namelists/scripts. You will learn more about this in future tutorials.

We are running a Startup case, therefore, the CONTINUE\_RUN option is set to FALSE.

```

#!/bin/csh -f

# Documentation of following environment variables is provided in env.readme

setenv RESUBMIT 0

setenv CCSMROOT /fis/cgd/cseg/csm/collections/ccsm3_0_1_beta19
setenv CASEROOT /home/bluesky/shields/ccsm3/scripts/b30.strat
setenv CONTINUE_RUN FALSE
setenv SETBLD AUTO # [AUTO, TRUE, FALSE]
setenv STOP_OPTION nmonths # [ndays,nmonths,daily,monthly,yearly]
setenv STOP_N 1
setenv BEST_OPTION $STOP_OPTION
setenv BEST_N $STOP_N
setenv INFO_DEBUG 1 # [0,1,2,3]
setenv DEBUG FALSE # [TRUE, FALSE]
setenv CHECK_TIMING TRUE # [TRUE, FALSE]
set OCN_TRACER_MODULES = (iage) # [(iage), (iage cfc), (cfc), ( )]

#=====
# DERIVED ENVIRONMENT VARIABLES (user may edit these)
#=====

# BUILD TYPE

if ($SETBLD =~ AUTO*) then
    setenv BLDTYPE TRUE
    if ($CONTINUE_RUN == "TRUE") setenv BLDTYPE FALSE
else
    setenv BLDTYPE $SETBLD
endif

# COUPLER HISTORY/DIAG NAMELIST SETTING

setenv HIST_OPTION monthly
setenv HIST_N -999 # unused
setenv HIST_64BIT .false. # use 32-bit (default behavior)
setenv AVHIST_OPTION monthly
setenv AVHIST_N -999 # unused
setenv DIAG_OPTION ndays
setenv DIAG_N 10 # current favorite value

setenv LOGDIR "" # save extra copy of log files here

#=====
# DERIVED ENVIRONMENT VARIABLES (user should not edit these)
#=====

setenv CONTINUE_RUN `echo $CONTINUE_RUN | tr '[a-z]' '[A-Z]`
setenv SETBLD `echo $SETBLD | tr '[a-z]' '[A-Z]`
setenv DEBUG `echo $DEBUG | tr '[a-z]' '[A-Z]`
setenv UTILROOT $CCSMROOT/scripts/ccsm_utils # root directory for csm script utilities
setenv CODEROOT $CCSMROOT/models # model source code root directory
setenv CSMDTL $CCSMROOT/models/utills # model util code root directory
setenv SHAREROOT $CCSMROOT/models/csm_share # model code shared code directory
setenv BLDROOT $CCSMROOT/models/bld # makefiles are here

```

# env\_run

## C.1.5 How to Run a “Startup” Case: Scripts

### 4) Configure

run the configure utility.....

```
configure -mach bluesky
```

### 5) Edit the b30.strst.bluesky.run script

- a) #@ wall\_clock\_limit = 3500
- b) #@ class = ded1\_rg8
- c) #@ account\_no = 54040010

## C.2. How to Run a “Startup” Case: Build

- 1) In your \$CASEROOT, run your buildscript... b30.strst.bluesky.build. This will build the model and preposition all the initial dataset.
- 2) While the model is compiling, take a moment to explore your /ptmp directories.
- 3) Let's also look at how Bluesky's queuing system works. The model is NOT run interactively from the command line, (as in the buildscript), instead, we submit a our runscript to the **batch queues**.

## C.3.1. How to Run a “Startup” Case: Run

Before we submit the model to the supercomputer, let's look at bluesky queues...

**llq**

or

**batchview**

csl\_pr8

com\_pr8

csl\_rg8

com\_rg8

csl\_ec8

com\_ec8

We will be running in a special queue on the 8-way nodes called ded1\_rg8 during the tutorial. On evenings and weekends, we will run in com\_rg8.

Bluesky also has queue for 32-way nodes.

Load leveller is what controls the queues on Bluesky

Useful loadleveler commands (II)....

Submit a run: `llsubmit <scriptname>`

Kill a run : `llcancel <job id>`

Where the job id can be found by typing “llq” and looking at the first column. The job id begins with the letters “bs” and is followed by a series of numbers and letters.

## C.3.2. How to Run a “Startup” Case: Run

- 1) Go to your \$CASEROOT directory
- 2) Type and enter `lsubmit b30.strst.bluesky.run`
- 3) Type and enter `llq | grep <logname>`
- 4) When the model starts running, `cd` to your `ptmp` directories...  
explore your `ptmp` runtime dirs and archive dirs.

For your runtime dirs, the following unix command is useful for viewing a layer of directories... `ls -ldart */*`

- 5) When the model is done running, look in your short term archiving directory at the various model output files. We will learn how to look at these later in this tutorial. (`ncview` and `ncl`).

## C.3.3. How to Run a “Startup” Case: Run

While the model is running we can look at the component model log files...

atm.log.yymmdd.ttttt

lnd.log.yymmdd.ttttt

ocn.log.yymmdd.ttttt

cpl.log.yymmdd.ttttt

ice.log.yymmdd.ttttt

where yymmdd.ttttt is a time stamp label for the log file

The log files contain the standard out from each component model. These log files are in the runtime dirs while the model is running, then after model completion, are moved to the short (and eventually long) term archiving directories.

## C.3.4. How to Run a “Startup” Case: Run

In your \$CASEROOT directory, you will also find a general standard output file and a standard error file.

poe.stdout.xxxxxxx.x

poe.stderr.xxxxxxx.x

Standard out and error not specific to individual component models can be found in these files.

## C.4.1. How to Run a “Startup” Case: Restart and Resubmit

- 1) Go back to \$CASEROOT
- 2) Edit env\_run,
  - a) change **CONTINUE\_RUN** to **TRUE**
  - b) change **RESUBMIT** to **1**

This tells the model to continue the startup run and run one more month.

The integer value for RESUBMIT tells the model how many times to resubmit the model after the job completion.

How many months total will the model run after we lsubmit the runscrip?  
How many months total will b30.strst have run (startup and restart)?

Do we need to recompile after editing env\_run for our resubmit case?

## C.4.2. How to Run a “Startup” Case: Restart and Resubmit

No, we do not need to recompile after changing `CONTINUE_RUN` or any `STOP_*` or `REST_*` option.

- 3) Edit **env\_mach.bluesky** file to allow the long term archvier to run. For our initial startup case, we did not turn on the long term archiver. For our restart and resubmit case, we will set

**DOUT\_L\_MS to TRUE**  
**DOUT\_L\_MSPRJ to 54040010**

- 4) Edit the **b30.strst.bluesky.l\_archive** for the appropriate project number. If no account number is given, your default project will be charged.

**#@ account\_no = 54040010**

- 5) `lsubmit b30.strst.bluesky.run`

## C.4.3. How to Run a “Startup” Case: Restart and Resubmit

How does the model know what files to restart from?

Look in your short term archiving “restart” subdirectory. You will find “rpointer” files that tell the model where to find the appropriate restart files.

## D. How to Run a “Hybrid” Case

We will now run a one month hybrid case.

How do we start?

## D.1.1. How to Run a “Hybrid” Case: Scripts

- 1) Go to our `$CCSMROOT` scripts directory and `create_newcase` using `b30.hyb` as our case name.
- 2) `create_newcase -case /home/bluesky/<logname>/ccsm3/scripts/b30.hyb -mach bluesky -res T31_gx3v5 -compset B -ccsmroot /fis/cgd/cseg/csm/collections/ccsm3_0_1_beta19`
- 3) Cd to our `$CASEROOT` directory and edit `env_conf`

What do we change in `env_conf`?

## D.1.2. How to Run a “Hybrid” Case: Scripts

```
RUN_TYPE      hybrid
RUN_STARTDATE 0001-01-01
RUN_REFCASE   b30.031
RUN_REFDATE   0400-01-01
```

Where the reference case, b30.031, is a T31\_gx3v5 CCSM3 Present Day Control Run. We will use year 400 to initialize our hybrid case.

# env\_conf

```
#!/bin/csh -f

# Documentation of following environment variables is provided in env.readme

setenv CASE b30.hyb
setenv CASESTR b30.hyb
setenv COMP_ATM cam
setenv COMP_LND clm
setenv COMP_ICE csim
setenv COMP_OCN pop
setenv COMP_CPL cpl
setenv CSIM_MODE prognostic
setenv GRID T31_gx3v5
setenv RUN_TYPE      hybrid          # [startup,hybrid,branch]
setenv RUN_STARTDATE 0001-01-01      # [yyyy-mm-dd]
setenv RUN_REFCASE   b30.031         # [only applies to hybrid or branch runs]
setenv RUN_REFDATE   0400-01-01      # [only applies to hybrid or branch runs]
setenv IPCC_MODE     OFF             # [OFF, 1870_CONTROL, 1870_TO_PRESENT, RAMP_CO2_ONLY,
                                     # FUTURE_A1, FUTURE_A2, FUTURE_B1, FUTURE_B2]
setenv RAMP_CO2_START_YMD 00000000   # Start date of CAM CO2 ramp in form YYYYMMDD
                                     # This variable MUST be set if IPCC_MODE is RAMP_CO2_ONLY
                                     # This variable is ignored if IPCC_MODE is not RAMP_CO2_ONLY

#=====
# DERIVED ENVIRONMENT VARIABLES (user should not edit these)
#=====

setenv ATM_GRID      `echo $GRID | sed s/._\*//`; setenv LND_GRID $ATM_GRID
setenv OCN_GRID      `echo $GRID | sed s/.\*\*//`; setenv ICE_GRID $OCN_GRID
```

## D.1.3. How to Run a “Hybrid” Case: Scripts

- 4) Edit `env_run`. We want to run for one month, no restarts, no resubmits... You can choose whether or not to include monthly coupler history files.

```
CONTINUE_RUN FALSE
SETBLD          AUTO
STOP_OPTION     nmonths
STOP_N          1
```

# env\_run

```
#!/bin/csh -f

# Documentation of following environment variables is provided in env.readme

setenv RESUBMIT      0          # if RESUBMIT is > 0, then will resubmit

setenv CCSMROOT     /fis/cgd/cseg/csm/collections/ccsm3_0_1_beta19
setenv CASEROOT     /home/bluesky/shields/ccsm3/scripts/b30.hyb
setenv CONTINUE_RUN FALSE      # [TRUE, FALSE]
setenv SETBLD       AUTO       # [AUTO, TRUE, FALSE]
setenv STOP_OPTION  nmonths    # [ndays,nmonths,daily,monthly,yearly]
setenv STOP_N       1
setenv REST_OPTION  $STOP_OPTION
setenv REST_N       $STOP_N
setenv INFO_DEBUG   1          # [0,1,2,3]
setenv DEBUG        FALSE     # [TRUE, FALSE]
setenv CHECK_TIMING TRUE      # [TRUE, FALSE]
set OCN_TRACER_MODULES = (iage) # [(iage), (iage cfc), (cfc), ( )]

# =====
# DERIVED ENVIRONMENT VARIABLES (user may edit these)
# =====

# BUILD TYPE

if ($SETBLD =~ AUTO*) then
    setenv BLDTYPE TRUE
    if ($CONTINUE_RUN == 'TRUE') setenv BLDTYPE FALSE
else
    setenv BLDTYPE $SETBLD
endif

# COUPLER HISTORY/DIAG NAMELIST SETTING

setenv HIST_OPTION  monthly
setenv HIST_N       -999      # unused
setenv HIST_64BIT   .false.   # use 32-bit (default behavior)
setenv AVHIST_OPTION monthly
setenv AVHIST_N     -999      # unused
setenv DIAG_OPTION  ndays
setenv DIAG_N       10       # current favorite value

setenv LOGDIR       ""        # save extra copy of log files here

# =====
# DERIVED ENVIRONMENT VARIABLES (user should not edit these)
# =====

setenv CONTINUE_RUN `echo $CONTINUE_RUN | tr '[a-z]' '[A-Z]`
setenv SETBLD       `echo $SETBLD | tr '[a-z]' '[A-Z]`
setenv DEBUG        `echo $DEBUG | tr '[a-z]' '[A-Z]`
setenv UTILROOT     $CCSMROOT/scripts/ccsm_utils # root directory for ccsn script utilities
setenv CODEROOT     $CCSMROOT/models # model source code root directory
setenv CSMUTIL      $CCSMROOT/models/utils # model util code root directory
setenv SHAREROOT    $CCSMROOT/models/csm_share # model code shared code directory
setenv BLDROOT      $CCSMROOT/models/bld # makefiles are here
```

## D.1.4. How to Run a “Hybrid” Case: Scripts

- 5) Run configure

```
configure -mach bluesky
```

- 6) Edit your env\_mach.bluesky script to activate long term storage

```
DOUT_L_MS and DOUT_L_MSPRJ
```

- 7) Edit your b30.hyb.bluesky run scripts to specify queue, wallclock, and account numbers.

```
#@ account_no  
#@ class  
#@ wall_clock_limit
```

- 8) Edit your b30.hyb.l\_archive script to specify account number

```
#@ account_no
```

## D.2. How to Run a “Hybrid” Case: Build and Run

- 1) Build the model from your \$CASEROOT directory by typing and entering **b30.hyb.bluesky.build**.

Why do we set the SETBLD to “AUTO” in the env\_run file?

- 2) After the model is built, and we check our /ptmp directories to make sure the build was successful and all the executables are present, submit the run script to the batch queues on bluesky.

**llsubmit b30.hyb.bluesky.run**

- 3) Check the queues to make sure are job was queued...

**batchview**

# REVIEW

## RUNS:

Startup/Restart	3 months total	b30.strst
Hybrid	1 month total	b30.hyb

We ran the long term archiver for our restart case. This script should have archived everything in our short term archiving directory, /ptmp/<logname>/archive/b30.strst. All 3 months of data should now be on the Mass Storage System (MSS).

We also ran the long term archiver for our hybrid case, so, consequently, one month of data for b30.hyb should be on the MSS.

How do we find this data?

# MASS STORAGE SYSTEM (MSS)

Key commands:

`msls -l <full pathname for file>` : shows to screen specified file

`msrcp` : copies files to and from mss

The MSS pathname convention for output files is as follows:

`/<LOGNAME>/csm/<CASE>/<model>/hist/<filename>` history files

`/rest/` restart files

`/init/` initial files

`/logs/` log files

See the CCSM3 user documentation (or simply explore your short and long term archiving) for details on all output file naming conventions.

# Examples:

Let's do an msls listing for an ice history file from our b30.strst case...

```
msls -l /SHIELDS/csm/b30.strst/ice/hist/b30.strst.csim.h.0001-01.nc
```

```
ice history file: b30.strst == casename  
                  csim      == model component  
                  h         == history file  
                  000101    == year1 month 1 (this is a monthly avg file)  
                  nc       == netcdf file
```

To look at an entire directory tree, type/enter

```
msls -l /<LOGNAME>/csm/b30.strst/<model> (no "/" at the end)
```

```
msls -l /<LOGNAME>/csm/b30.strst/<model>/rest
```

Take a moment to explore your mss listings... pick history, restart, or logs files.

# Examples:

Let's say that the long-term archiver has written all your model files to the MSS and has removed all but the last set of files produced by your most recent batch job.

Let's also say that you want to look at an ocean history file that is no longer on the short term archiving space.

How do you do this?

- 1) Make a work directory unrelated to your model directories... mkdir  
/ptmp/<logname>/work
- 2) msrccp mss:/<LOGNAME>/csm/b30.strst/ocn/hist/b30.strst.pop.h.0001-01.nc  
/ptmp/<logname>/work/b30.strst.pop.h.0001-01.nc
- 3) A short cut would be to cd into /ptmp/<logname>/work and type/enter  
msrccp mss:/<LOGNAME>/csm/b30.strst/ocn/hist/b30.strst.pop.h.0001-01.nc .

## MSS (continued):

### Documentation:

<http://www.cisl.ucar.edu/docs/mss>

<http://www.cisl.ucar.edu/docs/mss-commandlist.html>

### msls:

<http://www.cisl.ucar.edu/docs/mss/list.html#msls>

### msrcp:

<http://www.cisl.ucar.edu/docs/mss/read-write.html#msrcp>

## E. How to View Your Output

All of our data processing and viewing will be done on a different machine, an SGI, called tempest.

Please log on to tempest using  
`ssh -Y roy.ucar.edu ....`

# E.1. How to View Your Output: Netcdf

## NETCDF

All history output files are in “netcdf” format. (Other data formats include binary, ascii, etc).

Netcdf stands for “network Common Data Form”, and is a self-describing dataset. This means the file includes information about the data it contains... this information is called “metadata”.

We will be using several tools to manipulate and view netcdf data. The computer libraries required to utilize netcdf and it's tools are already installed on tempest and bluesky.

Documentation on the Netcdf format can be found on the Unidata website:  
<http://www.unidata.ucar.edu/software/netcdf/docs/>

# E.2.1. How to View Your Output: Ncview and ncdump

## **NCDUMP**

Documentation for ncdump can be found in the netcdf user's guide. There is a link to the user's guide on the Unidata website. The explicit link is:

<http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/>

Ncdump is a netcdf utility that allows one to dump the contents of the netcdf file to screen or file.

<http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/ncdump.html>

Files are often too big to dump to screen, but we can look at subsets of the file using the different ncdump options. Go to the above address to view these options.

Lets look at the metadata for one of our history files.

1. `cd /bsptmp/<logname>/archive/b30.hyb/ocn`
2. `ncdump -h b30.hyb.pop.h.0001-01.nc | more`
  - a. the “-h” option tells ncdump to only dump the header record out to the screen.
  - b. the “ | more” allows you to view the contents of the dump one screen page at a time.

Lets look at a variable from one of our history files.

1. `ncdump -v z_t b30.hyb.pop.0001-01.nc`
2.
  - a. the “-v” option tells ncdump to dump only the variable `z_t` to screen in addition to the meta-data.
  - b. the variable `z_t` contain the ocean layer depths from surface to midpoint of the layer, in centimeters.

## E.2.2. How to View your Output: Ncview and ncdump

### NCVIEW

Ncview is a graphical interface which allows us to quickly view the variables inside our netcdf file. This is not an ascii-like dump to screen, like ncdump, but rather a plotting tool.

Let's look at our of our model history files with ncveiw.

1. `cd /bsptmp/<logname>/archive/b30.hyb/lnd`
2. `ncview b30.hyb.clm2.h0.0001-01.nc`
3. Click on to a variable name to see the contoured data. Use a variable name with capital letters.
4. To change the contour intervals, click onto the "Range" box
5. In your spare time, you may want to play with this tool. It is very useful for "quick" history output data checks.

## E.3.1. How to View Your Output: NCL

NCL is the NCAR Command Language. It is an interpreted language designed for data processing and visualization.

<http://www.ncl.ucar.edu>

This language does take a bit of time to learn and use proficiently. For the purposes of this class, we will teach you to use 5 simple ncl scripts designed to specify and plot variables from model history files.

If you are interested in learning more about ncl, or are interested in using this language for data processing or visualization, go to the ncl homepage and click on “Training” to read about your options.

## E.3.2. How To View your Output: NCL

1. Go to the wiki and download the following ncl scripts:
  - a. atm\_latlon.ncl
  - b. ice\_north.ncl and ice\_south.ncl
  - c. ocn\_latlon.ncl
  - d. lnd\_latlon.ncl
2. Place these scripts in your home directory on tempest. Create a new subdirectory for these files called “nclscripts” if you like.
3. Each of these scripts is designed to plot a user specified field for the given model. You will need to do the following to each script:
  - a. edit the script in the USER DEFINED VARIABLES section to specify things such as history filename (case, path, etc.), field, time index (if applicable), and level index (if applicable).
  - b. On tempest, type “ncl <script.ncl>”, where script.ncl is the actual scriptname.

## E.3.3. How to View Your Output: NCL

### EXAMPLE: atm\_latlon.ncl

1. infile will be the full pathname where your history data resides.... ,i.e.,  
/bsptmp/<logname>/archive/b30.hyb/atm/b30.hyb.cam2.h0.0001-01.nc
2. field = "T" , if you want to look at Temperature. Temperature is a 3d field, so...
3. level\_index = 25 , where 25 is the bottom-most atmosphere model level.

4. time\_index = 0, this is a monthly history file, so there should only be one timeslice.

Note that ncl uses the indexing convention where the 1<sup>st</sup> array slice is 0 and the last array slice is n-1.

5. On tempest, in the directory where your atm\_latlon.ncl script resides,  
type "ncl atm\_latlon.ncl"

6. This script produces an postscript file. To view the postscript file type, "**ghostview** II\_T\_time31\_lev992.ps", where T\_time31\_lev992.ps is the postscript filename generated by atm\_latlon.ncl.

Now try running the other component ncl scripts!

# F.1. How to Process Your Output: NCO

NCO stands for netcdf Operators.

NCO is a suite of programs designed to perform certain “operations” on netcdf files, i.e., things like averaging, concatenating, hyperslabbing, or metadata manipulation.

If NCO is installed on your machine, these operations are performed on the command line, in a UNIX-type environment.

Command-line operations are extremely useful for processing model data given that modellers often work in a UNIX-type environment.

The NCO Homepage can be found at <http://nco.sourceforge.net>

The Operator Reference Manual can be found at:  
<http://nco.sourceforge.net/nco.html#Operator-Reference-Manual>

## F.2. How to Process Your Output: NCO

Common NCO programs you might use include:

**ncra** == netcdf record averager

Example: **ncra file1.nc file2.nc avgfile.nc**

file1.nc = input model history file, for jan year 1

file2.nc = input model history file, for feb year 1

avgfile.nc = new file consisting of jan/feb averaged data for all fields found in the input model history file.

## F.3. How to Process Your Output: NCO

**ncrcat == netcdf Record Concatenator**

**Example: `ncrcat file1.nc file2.nc file12.nc`**

file1.nc = input model history file, jan year 1

file2.nc = input model history file, feb year 1

file12.nc = new model history timeseries file  
consisting of the months of jan and feb, year 1.  
Each field in this file now has 2 timeslices.

## F.4. How to Process your Output: NCO

**ncks** == netcdf “Kitchen Sink”

“Kitchen Sink” is an American expression meaning “it includes just about everything”. This operator combines various netcdf utilities that allow you to cut and paste subsets of data into a new file.

Example: **ncks -v FIELD file1.nc FIELD\_file1.nc**

file1.nc = input monthly history file

FIELD = variable you wish to extract

FIELD\_file1.nc = the new netcdf file which contains only FIELD

i.e, if you wanted to create a netcdf file with only surface temperature, from an atmosphere model history file, case b30.hyb, you would do the following:

```
ncks -v TS b30.hyb.cam2.h0.0001-01.nc b30.hyb.cam2.TS.0001-01.nc
```

## G.1 Exercises: Building the Model

Build 4 Hybrid Model Cases.... These will be used in future tutorials.

Use b30.031 as your reference case.

Use year 400 jan 1 as your reference date

Name them:

b30.ACMatm1

b30.ACMInd1

b30.AC Mocn1

b30.AC Mice1

## G.2 Exercises: MSS, NCO

Using NCO operators, create seasonally averaged files (DJF, MAM, JJA, SON) from one year of data for the following case found on the MSS:

`/SHIELDS/csm/b30.hyb`  
(local dir `/bsptmp/shields/archive/b30.hyb`)

Do this for any model.

Do this in your `/ptmp` space on tempest.

## G.3 Exercises: NCO, ncview, ncdump

Using NCO operators, and using the same case as the Exercise 2, create an annually averaged file with only one variable.

For example, if you are using CAM data, create an annually average file with TS as the only variable on the file.

Use Ncview to look at the file.

Use ncdump to look at the header record for this file.

## G.4 Exercises: NCO, ncview, ncdump

Using NCO operators, and using the same case as in Exercise 2 and 3, create a 12 month timeseries file for the model of your choosing.

Again, work on your /ptmp space on tempest.

Use ncdump to view how the metadata changes when performing an NCO operation.

Use ncview to view the timeseries by clicking on the arrow buttons after choosing a variable.

## G.5 Exercises: NCL

Using the ncl script appropriate for your choice of model, plot a variable from a seasonal or annual averaged file. Play with changing the contour intervals. (The atm\_latlon.ncl script gives you an example on how to do this task). If you like, re-run the script for several variables of your choosing.

Try using the ncl script to plot different timeslices from your 12 month timeseries file.

If you are feeling adventurous, navigate the ncl web pages and try modifying your ncl script to read in your timeseries file and *average* the specified field before it is plotted. This will entail modifying code not in the USER DEFINED PARAMETERS section.

NCL is also a processing tool as well as a visualization tool... you can use either nco or ncl to manipulate your data. After you have been working with model data for a bit of time, you will find that some tasks are easier in nco while others are easier in a programming language, such as ncl.

REMEMBER to save your original scripts to someplace other than your working area (or save them under some another name) before modifying your scripts. Example, “cp atm\_latlon.ncl atm\_latlon.original.ncl”. This way, if you make mistakes, you can always go back to the original file and more easily start over again.