Chapter 3: A self-guided tour through the EMS

Chapter Contents:

3.1 EMS: What’s in this thing?
3.2 The EMS Tour: A view from the tippy top
3.3 The all-important EMS.cshrc and EMS.profile files
3.4 This is your EMS - And this is your EMS on Hyper-threading - Any questions?
3.5 All the environment variables you hate to love
3.6 EMS run-time routines
3.7 The many utilities you would have missed had they not been listed here
   3.7.1 Some additional EMS utilities
   3.7.2 Non EMS-developed tools and utilities

3.1 EMS: What’s in this thing?

At this point, you should have successfully installed the EMS on your computer, because if you haven’t, then this chapter will be of little use to you. So, if the EMS is not yet installed then go back to Chapter 2 and return only after you have completed that task.

At first glance it may appear that the EMS consists of numerous directories, subdirectories, and files that all require the constant attention of the user. The reality is that while there are a myriad of directories and files, users for the most part will not have to modify the contents. The EMS was designed so that both the novice and well-seasoned NWP user could feel comfortable with the system. Nearly everything comes pre-configured for most applications so you can just get down to business. Nonetheless, there are always exceptions, and thus this chapter provides an overview of the system should you feel it necessary to dig deeper into the bowels of the EMS.

3.2 The EMS Tour: A view from the tippy top

There are a number of environment variables that may be used to navigate the EMS, many of which will be explained in this chapter. The most important variable is $EMS, which defines the top level of the EMS installation. As you recall from Chapter 2, one test of a successful installation is the ability to change to the $EMS directory, i.e., “cd $EMS”. Go ahead and try it now, as this is where your self-guided tour will begin.

From the top level of the EMS you will see the following directories and files:

bin The bin directory contains the primary 32- or 64-bit WRF binaries installed with the EMS. These binaries have been specially selected for use with your system based upon the CPU type detected. If you are unable to appreciate the gift that the EMS developer has given to you and desire to compile your WRF binaries locally, then that can be accommodated. Please check out the contents of the util/wrfems directory or contact the SOO STRC; otherwise, you will fail.
conf     The **conf** directory default configuration files for the run-time routines and GRIB info files. Whenever you create a new domain, these files are copied into the local `<domain>/conf/<routine>` directory. If you continue to investigate **conf** you will see the following subdirectories:

- **ems_auto**     Contains the default configuration files for the ems_autorun.pl routine
- **ems_install**  Contains the configuration files for the ems_install.pl routine
- **ems_post**     Contains the default configuration files for the ems_post.pl routine
- **ems_prep**     Contains the lone configuration file for the ems_prep.pl routine
- **ems_run**      Contains the default configuration files for the ems_run.pl routine
- **grib_info**    Contains the configuration files for all the data sets used by ems_prep. There is lots of good reading here.

data    The **data** directory is the location for the large terrestrial data sets, default model configuration files, default namelist files, and tables used when using the system. There are quite a few files and directories beneath data; however, you probably won’t need to concern yourself too much with the contents unless you are trying to make changes to the system defaults.

Here is a brief description of each subdirectory and its contents:

- **domains**     Contains sample NMM and ARW domains - used with ems_domain.pl
- **geog**        Contains the various terrestrial data sets used with the WRF
- **tables**      Contains the default EMS configuration files and tables

docs     The **docs** directory contains assorted pieces of documentation on the WRF and EMS. Not all of the information is relevant to the EMS but users may find some value in the resources.

domwiz    The **domwiz** directory contains the Java libraries and binaries used when running the WRF Domain Wizard (DW).

logs     The **logs** directory contains the installation and Domain Wizard (dwiz) log files. It is also the default location for the ems_autorun.log file. The log files from the individual domain simulations are not located in “wrfems/logs” but can be found in “wrfems/runs/<domain>/log”.

runs     The **runs** directory is where all your magic happens. This is where your newly created domain directories will reside after you have completed a successful localization with the Domain Wizard. It is also where you run your simulations with the EMS.

strc     The **strc** directory and subdirectories contain all of the Perl routines and modules used to power the EMS. It is unlikely that you will ever need to edit the files in this directory. These files are also the most likely to be updated with each new EMS release.
The util directory contains many of the utility programs and packages provided with the EMS that help to put the “S” in “EMS”. Hint, the “S” is for “System”. The preconfigured domains used for running the benchmark cases reside here as well as NAWIPS and GrADS visualization packages. There are also additional files and directories that may be of interest to you so it is worth spending some time investigating all that “wrfems/util” has to offer.

Here is a brief description of the util subdirectory contents:

- **arwpost**: The binaries and namelist files for running arwpost
- **bin**: Various precompiled non-EMS developed utilities (Section 3.6.1)
- **benchmark**: The NMM and ARW core benchmark cases. This is where you run the benchmarks that are described in Appendix B.
- **grads**: GrADS binaries, scripts, and data files
- **mpich2**: MPICH2 binaries and documentation
- **nawips**: NAWIPS model data manipulation and display package
- **ncview**: NcView binaries, scripts, and data files
- **workshop**: Contains the data, html files, and documentation for EMS lab exercises
- **wrfems**: The EMS-modified WPS and WRF release files. These are the files that you will want to include when building the WRF or WPS locally from scratch.
- **wrfutils**: Various precompiled NCAR-provided WRF binaries

### 3.3 The all-important EMS.cshrc and EMS.profile files

When you log in as EMS user, a number of environment variables are set that are used by the Perl modules and executables included with the system. All of these variables are defined in the EMS.cshrc and EMS.profile files, for (T)Csh or bash users respectively, which reside in the etc directory ($EMS/etc). The user, if necessary, may modify some of the variables in this file, but for the most part the EMS install tool should have done an adequate job during the configuration process. Nonetheless, it’s always good to make sure you are happy with the settings.

Here is a summary of the environment variables you can modify should you want to feel empowered:

**NCPUS, CORES and OMP_NUM_THREADS**

The NCPUS environment variable defines the number of physical processors that reside on your local system. NCPUS **does not** define the total number of processors (NCPUS * CORES), nor does it define the processors used when running a simulation. The NCPUS variable is simply the number of actual CPUs that you could touch if you were to open up the computer case and move the large heat sinks. Most stand-alone workstations have either 1 or 2 physical CPUs although some exotic and expensive mainboards can support 8 or more, but you probably don’t have one of those babies.

The CORES environment variable defines the number of cores contained on each of those touchable CPUs. The description of the processor type in the /proc/cpuinfo file might provide a clue as to the number of cores on each CPU such as “dual core”, “Duo”, “quad core”, or “6-Core”. If you read a “tri-
core” or “hepta-core” then you got other problems. If you really don’t know how many cores there are on each CPU then try running the EMS “sysinfo” utility; otherwise, look up your CPU on that Internet thing.

The NCPUS and CORES environment variables are used by the EMS to define the total number of processors available on your system (NCPUS * CORES), the value of which is stored by the OMP_NUM_THREADS environment variable. It is important to have these values correct as many of the binaries are compiled to run on multiple processors and the EMS needs to know just how many processors are available. Note that the EMS can easily be configured for distributed computing across multiple Linux systems, but only for the “emsupp”, WRF “real.exe” and the NMM and ARW core executables. Other binaries are compiled for distributed computing on the local host while still others run in shared memory. You cannot, for example, distribute the processing of initialization data across a cluster of workstations. This minor limitation was introduced to simplify EMS development since the advantage in exporting computational load to other systems was minimal.

Also note that setting either NCPUS or CORES greater than the true values for your system will result in degradation of EMS performance. You can do it, and the Linux kernel will comply by creating the requested number of threads, but your performance will suffer, and you don’t want that to happen.

Finally, don’t get all giddy with excitement if the EMS installation tool indicates that you have twice as many physical CPUs and cores on your computer than you think it does. If hyper-threading is turned ON in the BIOS then the system can incorrectly report the number processors and cores by 2-fold. In that event you will have to manually assign the correct values to NCPUS and CORES in EMS.cshrc (or EMS.profile) as your performance will be compromised. You are better off turning hyper-threading OFF in your BIOS but that is up to you. Please read Section 3.4 for an explanation - multiple times.

**DSKCHEC**

The DSKCHEC environment variable allows the user to monitor disk space usage on the partition where the EMS resides. Running an NWP model can generate lots and lots of Giga-chunks of data and, if not monitored, a partition can fill up with less-than-desirable consequences. Setting DSKCHEC to "Yes" will result in the EMS checking available space before the start of each EMS run-time routine. If space is limited (greater than 95% usage), a warning message will be printed to the screen. The routine will also automatically terminate your simulation and provide you with a message (via email too) when your usage reaches 100%. That’s the way it’s supposed to work at least.

**MAILEXE**

The MAILEXE environment variable defines the mail routine to use when sending you informative messages or pearls of nonsensical wisdom, which tends to happen from time to time. Just like a fortune cookie that provides lottery numbers. Note that while the email recipients for disk space warnings are defined by the USERS variable, there is a separate USERS configuration available for each user-created domain located in the ems_autorun.conf configuration file. More information about that file is provided in Chapter 10.

**USERS**

The USERS environment variable provides a list of email addresses that will be used in the event of a disk space problem only (See DSKCHEC). This variable is not to be confused with the other USERS
parameter, available in each domain ems_autorun.conf configuration file, which allows you to send email to a separate list of users in the event of a failed model run. Similar, but different.

Commas separate individual email addresses. Leave USERS blank if you don’t want mail sent because it just gets in the way of your solicitations for “male enhancement” products.

**NAWIPS**

By default, the NAWIPS setting is commented out. You may remove the “#” if you wish to use the NAWIPS display package interactively; that is, from the command line, for generating images or displaying your model output. *You do not have to set the NAWIPS environment variable for generating BUFKIT files or images using the included script_drvr.csh routine as it will be done automatically.*

One consequence of including the NAWIPS environment is that it can increase the number of user-defined environment variables to a value greater than 343. Why is the number “343” important? Well, because there is an internal check in MPICH2 for the number of environment variables set. If a user has more than 350 (343+7 that MPICH2 includes) the MPICH2 routines will fail to start. I don’t know why. I didn’t write MPICH2 but it is there.

3.4 **This is your EMS - And this is your EMS on Hyper-threading - Any questions?**

I’ll probably have to repeat this diversion somewhere else in this guide but this is as good a place as any to cover this topic. In regards to the use of hyper-threading with the EMS, *I do not recommend using the additional threads available when hyper-threading is turned on in the BIOS.* The reason for this statement is that you are much more likely to see degradation in performance rather than improvement. Yes, there are a limited number of conditions under which you might see some improvement depending upon your processors, the amount of cache, your domain configuration and the decomposition; however, you are far more likely to see degradation in the performance.

Keep in mind that when you use hyper-threading, you are only increasing the number of threads available to a process. *You are not increasing the computational resources,* such as the total number of processors available. If the number of threads created exceeds the number of available processors, then multiple threads will be assigned to a single processor. Consequently, each thread will have to share the resources allocated to that processor, such as memory and cache. This is beneficial only if the total amount of resources needed by the multiple threads is less than that available on the processor and there is no sharing of resources between threads. It is the sharing of resources, specifically cache, that increases the overhead and degrades performance.

As an example, say you have a dual CPU, quad core system that has a total of 8 processors available to use. If you request that all 8 processors be included in your simulation the WRF model will decompose your domain into 8 patches, each of which is assigned by the kernel to a processor as a single thread. Since there is a single thread assigned to each processor, there is no sharing of available resources. The model/system handles all the communication between the patches.

If hyper-threading is turned on and you request that 16 processors be used, the model will decompose your domain into 16 patches and the kernel create 16 threads. The problem is that since you only have 8 processors there will be 2 threads assigned to each. Remember that each thread will likely have to share resources on the processor, specifically cache, and it’s the overhead involved in the sharing between
threads that leads to the degradation in performance.

3.5 All the environment variables you love to hate

There are additional environment settings that are defined in the EMS.cshrc and EMS.profile files that are used by the EMS routines and utilities; even a few of which that may be of use to you. Those variables that may be of interest are identified below:

$EMS The top level of the EMS installation (wrfems)
$EMS_BIN The primary directory where the EMS binaries reside (wrfems/bin)
$EMS_DATA The directory where the static data sets reside (wrfems/data)
$EMS_STRC The location of the EMS STRC Perl modules and routines (wrfems/strc)
$EMS_RUN The directory where all your computational domains are located (wrfems/runs)
$EMS_UTIL The utility (closet) directory (wrfems/util)
$EMS_CONF The location of the default configuration and GRIB information files (wrfems/conf)
$EMS_LOGS Location of the general EMS log files (wrfems/logs)
$EMS_DOCS Location of the EMS and other WRF documentation (wrfems/docs)

3.6 EMS run-time routines

The run-time routines are the heart and soul of the EMS, if it actually had a soul. These are the tools that you will need to acquire and process initialization data, run simulations, and post-process the model output files. They are also used to automate the entire process when running a real-time forecast system. Be kind to the run-time routines and they will be kind to you.

All the run-time routines and ancillary modules are located below the wrfems/strc ($EMS_STRC) directory; however, they should never be run from this location. Instead, when you create a computational domain (Chapter 5), symbolic links will be made from your domain directory to the files in wrfems/strc. When you look at the top level of your domain directory, you will see that the “.pl” has been removed in the link name so that “<command>” points to “<command>.pl” in wrfems/strc. Taking this approach makes it easier for the EMS to keep things tidy. And a tidy house is a happy house.

Each of the run-time routines will be discussed ad nauseam in subsequent chapters of this guide; however, now is a fine opportunity to introduce them and provide you with the opportunity to say “Hello.”

The primary (big) four (kind of like the “BCS” of the EMS):

- **ems_prep** Used to identify, acquire and process the external data sets for use as initial and boundary condition information in the EMS.
- **ems_run** Ingests the output from ems_prep, creates the initial and lateral boundary conditions for the model run, and then executes the simulation.
- **ems_post** Processes all the model output and exports the data files to exotic locations of your
Chapter 03 – A Self-Guided Tour Through The EMS

dreams.

**ems_autorun**  Automates the process of running ems_prep, ems_run, and ems_post in succession for the purpose of creating a real-time simulation experience.

There are two additional run-time utilities that you will likely use but are not part of the “EMS BCS”. They do not have links from the domain directories, but you can probably handle that.

**ems_clean**

The **ems_clean** routine is used to return your run-time domain directory to a user-requested state. This process includes the removal of unnecessary files and directories and reestablishing symbolic links. The routine is automatically run whenever you execute one of the “BCS” run-time routines above, but may also be run manually from within a run-time domain with the level controlled by the “--level #” flag, i.e.:

**USAGE:**  %  ems_clean --level 3

The levels of cleaning ranges from simple removal of log files and resetting links (level 0) to relocating your computational domain and resetting the configuration files to the default settings (level 6). Most users will employ levels 1, 3, or 4. Running “ems_clean --help” provides a more detailed explanation of the power that is ems_clean.

**ems_autopost**

The **ems_autopost** routine is called from within ems_run to start the processing of your model output concurrent with the running of the simulation. You will never start ems_autopost directly from the command line and may not even know it exists, but it does, just like the Keebler Elves and the next release of the EMS. This routine is most frequently used when running ems_autorun for real-time forecasting purposes. More details on ems_autopost can be found in Chapter 11.

3.7  The many utilities you would have missed had they not been listed here

The EMS includes a number of utilities that may be used to enhance your local modeling experience, and then there are some that are simply useless. Below is a summary of those utilities and the location where they reside.

3.7.1  Some additional EMS utilities

The following utilities are located in the wrfems/strc/ems_bin directory, but should be in your path when you log in so there is no need to point to them directly.
**benchtest**

The **benchtest** is used to run an ARW or NMM core benchmark case multiple times and return the average, maximum, and minimum amount of time to complete the simulations. This utility is useful when you want a more accurate estimate of performance differences between systems, as using the information from a single benchmark run may not be sufficient.

**USAGE:** `% benchtest <core> <number iterations>`

**cnet**

The primary purpose of **cnet** is to determine whether a system is reachable from the machine on which it is being run. The program takes either a host name or IP address and then skillfully checks whether the system is resolvable and currently reachable. If the answer to these questions is "Yes" then cnet spits out the hostname, IP, and network interface used to reach the remote system.

Returned values of 0 (host, ip, iface, [ssh]) indicate that there was a problem, which may be better illuminated by passing the '-v' flag.

**USAGE:** `% cnet [option] Hostname or IP`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ssh</td>
<td>Include a test for correctly configured passwordless SSH</td>
</tr>
<tr>
<td>--v</td>
<td>V</td>
</tr>
<tr>
<td>--help</td>
<td>Print out this menu again in case you missed it the first time</td>
</tr>
</tbody>
</table>

**ems_domain**

The primary purpose of the **ems_domain** routine is to create and/or localize a computational domain for use with the EMS. It may be used as an alternative to the domain wizard if the DW GUI is not available; however, using ems_domain requires that you understand the basics about model domain configuration and grid navigation, as there are no pretty depictions of your domain.

After executing ems_domain your domain should be ready to run simulations, with or without you.

**USAGE:** `% ems_domain [--info [domain] or --newdom <domain> or --localize [domain] or - -import <directory>] [Other Options]`
grib2cdf

The `grib2cdf` routine reads the contents of a GRIB file and writes out a netCDF-formatted file along with a companion “cdl” file. The netCDF file will have a name similar to that of the GRIB file, except with an ”.nc” extension.

**Usage:**

```
% grib2cdf <grib file> ... <grib file>
```

This utility is simply a front end to a variety of programs that are used to convert the data from one format to another and then extract the necessary information. There is nothing fancy going on here, just a lot of hard work by the EMS Oompa-Loompas.

gribnav

The `gribnav` utility is one of many tools that may be used to check the navigation of a GRIB file. This utility will provide the projection type, grid dimensions and spacing, true and standard Lat/Lon, the pole and corner points of the domain defined in a GRIB file. It should work with most all of the EMS grid navigation options, although feel free to point out any “short comings” to receive bonus points and valuable prizes.

**Usage:**

```
% gribnav <GRIB 1 or GRIB 2 file>
```

mpicheck

The `mpicheck` routine should be used prior to running the EMS on a cluster of Linux computers. It was designed to identify potential problems that may arise, as MPICH2 is rather picky about the networking and communication configuration in your cluster. To use, run mpicheck from the master node on your system:

**Usage:**

```
% mpicheck master node1 node2 node3 ... nodeN
```

Where “master” and “node#” are replaced with the hostnames of the machines in the cluster. If you are running the EMS on a single workstation, then don’t worry about it unless you are just looking for fun but want to be disappointed.
ncpus

The primary purpose of ncpus is to determine the number of available processors for the machine on which it is being run. It simply reads the contents of the /proc/cpuinfo file, does some magic and then spits out a number. Whether it is the correct number depends completely upon the skill of the author to distill the available information into something meaningful, such as "42"

A value of 0 (cpus) indicates there was a problem, which may be better illuminated by passing the '-v' flag.

**USAGE:** % ncpus [--v | --V | --verbose] [--cores]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--cores</td>
<td>Print out the number of Physical CPUs and cores per CPU on system</td>
</tr>
<tr>
<td>--v/V/verbose</td>
<td>Print out additional information for debugging</td>
</tr>
<tr>
<td>--help</td>
<td>Print out this menu again in case you missed it the first time</td>
</tr>
</tbody>
</table>

runinfo

The runinfo routine is used to collect details about a simulation configuration and print it out in a semi-comprehensible format for the user. It's run from the command line within a domain directory after ems_prep has been executed.

The only option to runinfo is "--domains" (see below), which specifies the nested domains to be included in the printed information.

**USAGE:** % runinfo [--domains 1,..,N]

sysinfo

The sysinfo utility is used to provide a summary of your computer system, including the hostname, IP address, Linux distribution and kernel, the number and type of processors on the machine, as well as some additional information. Here is an example of the information provided:

**USAGE:** % rozumal@seven-> sysinfo

Starting EMS routine sysinfo (V3.4.1.12.40) on seven at Thu Apr 4 20:30:33 2013 UTC

* Testing whether each compute node is reachable via Ping:

* Testing Passwordless SSH from seven

    Passwordless SSH to seven - Success
* Gathering information for localhost seven

--------------------------------------------------------

System Information for seven

System Date : Thu Apr  4 20:30:34 2013 UTC
System Hostname : seven
System Address : 128.117.107.45

System OS : Linux
OS Kernel : 2.6.32-220.el6.x86_64
Kernel Type : x86_64
Linux Distribution : CentOS release 6.4 (Final)

Network Interface Information for seven

Network Interface : eth0
Interface Address : 128.117.107.45
Address Resolves to : Nothing
Interface State : Up

Network Interface : eth1
Interface Address : None Assigned
Address Resolves to : Nothing
Interface State : Up

Network Interface : lo
Interface Address : 127.0.0.1
Address Resolves to : Nothing
Interface State : Up

Network Interface : virbr0-nic
Interface Address : None Assigned
Address Resolves to : Nothing
Interface State : Inactive

Processor and Memory Information for seven

CPU Name : Intel(R) Xeon(R) CPU X5670 @ 2.93GHz
CPU Instructions : nehalem
CPU Type : 64-bit
CPU Speed : 1600 MHz

EMS Determined Processor Count
Physical CPUs : 2
Cores per CPU : 6
Total Processors : 12

EMS.cshrc Specified Processor Count
Physical CPUs : 2
Cores per CPU : 6
Total Processors : 12

Hyper-Threading : Off
System Memory : 23.5 Gbytes

EMS User Information for rozumal on seven

User ID : 8010
Group ID : 1750
Home Directory : /home/comet/rozumal
Home Directory Mount : NFS
User Shell : /bin/tcsh
Shell Installed : Yes
Shell Login Files : .cshrc
EMS.cshrc Sourced : .cshrc
EMS.cshrc Port Range : None Defined

EMS Installation Information for seven

EMS Release : 3.4.1.12.40
EMS Home Directory : /usr1/wrfems
EMS Home Mount : Local
EMS User ID : 8010
EMS Group ID : 1750
EMS Binaries : x64

EMS Run Directory : /usr1/wrfems/runs
EMS Run Dir Mount : Local
EMS Run Dir User ID : 8010
EMS Run Dir Group ID : 1750

Run Dir Avail Space : 809.31 Gb
Run Dir Space Used : 53%
EMS Util Directory : /usr1/wrfems/util

Your awesome EMS sysinfo party is complete - Thu Apr 4 20:30:34 2013 UTC
The EMS Metaphysician says: "Think Globally, Model Locally!"

3.7.2 Non EMS-developed tools and utilities

Here is a brief summary of the “value added” utilities that are provided with the EMS, many of which are used by the run-time routines, so it’s in your best interest to not rename or remove them. Besides, a few may be of use to you:

Located in the wurfes/util/bin directory:

a) cnvgrrib  Converts between GRIB 1 and GRIB 2 formats (goes both ways)
b) copygb    Remaps GRIB 1 files to a new grid navigation/projection
c) cpuid     Provides information of about your CPU
d) g1print   Prints out information in GRIB 1 files for creating a Vtable
e) g2print   Prints out information in GRIB 2 files for creating a Vtable
f) gribdump  A GRIB 1 file interrogation tool
g) ncdump    A netCDF file interrogation tool
h) ncview    A netCDF file visualization tool
i) rdwrfin   Prints out a summary of WRF intermediate format files
j) rdwrfnc   Prints out a summary of WRF netCDF format files
k) wgrib     A very special GRIB 1 file interrogation tool (EMS-modified)
l) wgrib2    A very, very special GRIB 2 file interrogation tool (with EMS love added)