CSU - RAMS

Model and 3rd Party Software Installation and Compilation and Linux User Setup

This document contains detailed notes on how to setup the RAMS software and all the pre-cursor software needed to run the model. This also contains examples of Linux .bash_profiles for 32 and 64-bit systems given the suggested software versions of the Intel Fortran compiler, HDF5, openMPI, and MPICH.

Updated by:

Stephen Saleeby
Department of Atmospheric Science
Colorado State University

Last updated: 13 October, 2014

##

GENERAL SOFTWARE AND RAMS INSTALLATION INSTRUCTIONS

##

RAMS build procedures

The 'bin.rams' and 'bin.revu' directories contain the build system for RAMS and REVU. There is a single file to modify for compilation.

- include.mk - contains library definitions, compiler option settings, and list of object files.

*** Set your include.mk file with paths to your RAMS user root path, HDF5 libaries, fortran and C executables and/or libraries, and MPI libraries. Also make sure the RAMS version number is what you intend.

*** Make sure to recompile the entire code if any changes are made to the F90 and/or module files.

*** If everything is set up correctly, you can simply type 'make clean' for a cleanup of old files and then 'make' within bin.rams and within bin.revu to compile these codes, respectively. By default this creates executables named 'rams-<versionnumber>' and 'revu-<versionnumber>'

More details on this process are given below.

##
##

Installation instructions

Install external libraries

First, you will need FORTRAN and C compilers to proceed.
We have had success if PGI, INTEL, and GFORTRAN

There are 2 external software packages that should be installed on the machine on which RAMS is to be compiled. If you do not have these installed, follow the instructions in the sections below.

1) HDF5: (Do an internet search to find the latest release)

   RAMS files are now using HDF5 format. Download the pre-compiled libraries for your computer/compiler, or download the source and compile yourself.

   *** Only the C libraries are needed, not the Fortran interface.

   ##
   ### Note: a tested version of HDF5 is provided (see below) ###
   ##

2) MPI: (Do an internet search to find the latest release)

   If you do not have a parallel computer, you will want to compile
   a sequential executable (non-parallel) by NOT defining
   -DRAMS_MPI in your include.mk file when you compile (see below). You
   will not need MPI libraries.

   If you have a parallel machine to use, you will want to compile with
   MPI support. The instructions to compile in parallel will vary
   Depending on your platform. We have found that there are mostly two
   Ways to execute:

   a) If you are compiling on a shared-memory machine (e.g., Sun, IBM,
      SGI, HP, etc.) using the manufacturer's version of MPI, you will
      need to consult their documentation as to the location of the
      libraries and include files. These locations will be specified in
      the include.mk files.

   b) For most Linux machines (and many others), you will probably want to
      run the MPICH, MPICH2, or openMPI packages.

   *** Only the C libraries are needed, not the Fortran interface.

   ###
   ### Note: tested versions of openMPI and MPICH are provided ###
   ###

3) EXAMPLES OF SETTING UP SOFTWARE and .BASH_PROFILE are BELOW.

4) Note that in the full RAMS distribution we have included pre-compiled
   binaries and libraries for HDF5, MPICH2, and OPENMPI as well as the
   source code. It might be good to start with this software since we have
   tested these versions with success. Otherwise these are all open source
   software available for free download over the web.

Access surface characteristic datasets

There are several datasets that define surface characteristics that can be
used in a RAMS run. These are not strictly required; it is possible to
configure a run without them. However, the majority of applications will
be configured for a real world location, so datasets that define
topography, sea surface temperature, land use, soil type, and NDVI are
usually used. These data can be downloaded from the RAMS webpage. See the
RAMSIN namelist documentation on use of these datasets.

Access and extract tar files

Un-tar the distribution file into a top-level directory. This top-level
directory will be the RAMS_ROOT used in subsequent steps.

Compile RAMS

---

1) Edit the include.mk file. Several things in this file need to be changed:

   a) Change the RAMS_ROOT variable to point to the top-level directory and change the version number if necessary.

   b) HDF5 library location - set HDF5_LIBS to the correct load paths of your HDF5 installation. Set HDF5_INCS to the directory where the C include file 'hdf5.h' exists.

   c) Machine-dependent options - Several variables must be set:

      CMACH - name of machine/compiler type. Used in a few source files to conditionally compile code sections. Our include file and code currently only contains 'PC_LINUX1' option for Linux framework. If you have a very different platform, do a code search for 'PC_LINUX1' to see where you need to make modifications.

      The routines that use this system condition name are rsys.F90 and include/utils_sub_names.h. These are related to system specific command line argument collection, clock timing methods, and use or non-use of C subroutine underscores. You may need to examine these files and modify code for you system since PC_LINUX1 is our only currently tested platform.

      F_COMP - name of Fortran compiler

      F_OPTS - command line options for the Fortran compiler.
      DO NOT ASSUME THAT THE OPTIONS IN THE include.mk FILE ARE REQUIRED OR THE BEST OPTIONS, OR THAT THEY EVEN WORK!!!!!
      We do not have access to all machine types to test.

      C_COMP - name of C compiler

      C_OPTS - command line options for the C compiler

      LOADER - loader command to use for linking, usually the Fortran compiler.

      LOADER_OPTS - options for linking

      LIBS - any other libraries required for linking

      ARCHIVE - archive command. Typically 'ar rs'.

   d) MPI location - PAR_INCS, PAR_LIBS and PAR_DEFS need to be set correctly. If not using MPI, these can be set to blank. If using MPI, make sure PAR_DEFS=-DRAMS_MPI and PAR_INCS and PAR_LIBS are correct. One group in this section must be uncommented.

2) Compile with the following commands within bin.rams and/or bin.revu:
   make clean (to clean up old files for a fresh compile)
   make   (to compile)

   If all goes well, the executable 'rams-<versionnumber> will be created and/or 'revu-<versionnumber>.

Execute

NOTE: SOMETIMES RUNNING THE MODEL WILL GENERATE RANDOM SEGMENTATION FAULTS THAT CANNOT BE TIED TO A PARTICULAR LINE OF CODE FOR MISSED MEMORY ALLOCATION. RATHER THE COMPUTER STACK MAY BE OVERFLOWING. TO AVOID THIS YOU CAN INCREASE YOU LINUX SYSTEM STACK SIZE OR SET IT TO UNLIMITED. THERE SHOULD BE NO REAL PROBLEM WITH THIS. THE COMMAND FOR LINUX IS:

To see your current stack limit:

ulimit –a

To set your stack to unlimited:

ulimit –s unlimited

The basic RAMS command line (for a non-parallel run) is:

rams-exec -f <namelist_file>

where rams-exec is the rams executable and namelist_file is the file name of the input namelist file. The default namelist_file name is 'RAMSIN'.

THE SETTING OF VARIABLES IN THE DISTRIBUTED NAMELIST FILES SHOULD NOT BE CONSIDERED THE 'DEFAULT' VALUES. Each parameter in the namelist should be considered with every model configuration. See the Namelist doc.

The normal procedure for executing an ISAN/RAMS run is:

- Execute RAMS (RUNTYPE='MAKESFC') to generate the surface files
- Execute ISAN (RUNTYPE='MAKEVFILE') to generate the varfiles
- Execute RAMS (RUNTYPE='INITIAL') to produce the model simulation

---
Test RAMS

In the 'test' directory we have provided a test for you to use.

The script 'x.testruns.quik.sc' will attempt to perform a single test with a supercell thunderstorm setup using the RAMSIN file named 'x.RAMSIN.supercell'.

To perform this test, you will only need the Fortran and C compilers and HDF5 installed. The default setup is for a non-parallel simulation, so

MPICH or openMPI are not needed. This is meant to be a basic test to see if implementation was successful.

Lastly, note, that you can examine model fields directly with software that can directly open HDF5 files. Otherwise you will need to use the REVU post-processing package. This package will output additional derived quantities that the user chooses and can output in text format or HDF5. Note that IDL, MATLAB, and GRADS can all open HDF5 files. The default format for the text option, is in GEMPAK format so that this data can be directly input into a GEMPAK grid file.

##
Notes on a few specifics if you are contributing code to RAMS. We are starting to force some general coding practice to make doing code searches easier.

Note: all module, subroutine, and function names should be lower case and all calls or usage of them should be lower case. This makes for ease of direct 'grep' matching and script searching when examining code usage.

1.Head every module, subroutine, interface, and function with letter case -specific opener. This should begin at column1 and include the spaces and parentheses.

  for subroutine example: 'subroutine <routinename> ()'
  for module example: 'module <modname>'
  for interface example: 'interface'
  for function example: 'integer function <funcname>'

2.Use 'implicit none' for every program, subroutine, module, and function and place in column 1. The does not have to be added for 'entry' statements within subroutines.

3.End every program, module, subroutine, interface, and function with letter case specific closure:

  'End Program main'
  'End Subroutine <namehere>'
  'End Module <namehere>'
  'End Interface'
  'End Function <namehere>'

4. Module 'use' statements should be in column1 before 'implicit none'.

5. For calling subroutines, use 'CALL' in all capital letters, place a space after the name of the subroutine to call, and always use () after the name of the subroutine to call.
   (ie. 'CALL cldnuc ()')

6. When using a function in a routine, make sure to declare the function as an 'external' variable and make the function call name all lower case and put parentheses right after the name.
   (ie. 'real, external :: rslf')
   (ie. 'rslf()')
##

##

SETTING UP 3RD PARTY SOFTWARE ON LINUX

##
##

For getting RAMS ready to work on a Linux machine:
Used the following software:

---

1. Install you favorite fortran compiler. RAMS has been tested with PGI, INTEL, and GFORTRAN. However, RAMS runs slower with GFORTRAN.

   For using Intel fortran compiler (installed this first):
   ifort12.compxe_ia32_2011.8.273.tar.gz (for 32 bit)
   ifort12.compxe_ia64_2011.8.273.tar.gz (for 64 bit)

   a. run 'install.sh' and follow commands
   b. passcode serial number is 'XXXX-XXXXXXXX'
   c. in your .bash_profile, add
      for 32 bit:
      PATH=/opt/intel/composer_xe_2011_sp1.8.273/bin/ia32
      for 64 bit:
      PATH=/opt/intel/composer_xe_2011_sp1.8.273/bin/intel64

2.1 MPICH-2

   MPICH-2 SOURCE CODE to compile:

   mpich2-1.4.1.tar.gz (works for either 32 or 64 bit)

   a. run './configure --prefix=/usr/local/mpich2-1.4.1 --disable-f77 \
        --disable-fc --disable-cxx --enable-fast=O3 CPPFLAGS=-DNDEBUG \
        --with-device=ch3:nemesis CC=gcc'
      or keep it simple and try without command line flags.
      Might need to include or not-include the ch3:nemesis option
      depending on your system.
   b. run 'make'
   c. sudo in as root
      run 'make install'
   d. For other software to work with MPICH, keep MPICH located
      within the installed directory. If you need to change location, then
      you need to recompile to the new location.

   For either installation to run, you may need to update your environment
   as examples show below:

   a. In your .bash_profile, make sure the following are included:
      As an example:
      PATH=/usr/local/mpich2-1.4.1/bin
      LD_LIBRARY_PATH=/usr/local/mpich2-1.4.1/lib
   b. To run with 'n' processors, execute:
      'mpiexec -f machinefile -n <number> <executable'
      The 'machinefile' is of the form:
      host1

```
      host2:2
      host3:4   # Random comments
      host4:1
      'host1', 'host2', 'host3' and 'host4' are the hostnames of the
      machines you want to run the job on. The ':2', ':4', ':1' segments
      depict the number of processes you want to run on each node. If
      nothing is specified, ':1' is assumed.

-------------------------------------------------------------------------
2.2 OpenMPI (could install this instead of MPICH, but do not have to)

   OPENMPI SOURCE CODE to compile:

   openmpi-1.4.5.tar.gz (works for either 32 or 64 bit)

   a. To create with shared object libraries:
      run ‘./configure --prefix=/usr/local/openmpi-1.6.5 \
        --disable-mpi-f77 --disable-mpi-f90 --disable-mpi-cxx \
        CC=gcc CFLAGS=-O3’
      To create with static libraries:
      run ‘./configure --prefix=/usr/local/openmpi-1.6.5 \
        --disable-mpi-f77 --disable-mpi-f90 --disable-mpi-cxx \
        CC=gcc CFLAGS=-O3 --enable-static disable-shared
   b. run ‘make’
   c. sudo in as root
      run ‘make install’

   For either installation to run, you may need to update your environment
   as examples show below:

   a. In your .bash_profile, make sure the following are included.
      As an example:
      PATH=/usr/local/openmpi-1.6.5/bin
      LD_LIBRARY_PATH=/usr/local/openmpi-1.6.5/lib
   b. To run with 'n' processors, execute:
      ‘orterun -f machinefile -n <number> <executable>‘
      The 'machinefile' is of the form:
      host1
      host2 slots=2
      host3 slots=4   # Random comments
      host4 slots=1
      'host1', 'host2', 'host3' and 'host4' are the hostnames of the
      machines you want to run the job on. The 'slots=2', 'slots=4',
      'slots=1' segments depict the number of processes you want to
      run on each node. If nothing is specified, 'slots=1' is assumed.

-------------------------------------------------------------------------
3. HDF5 (installed this third):

   HDF5 PRECOMPILED Software with static or shared object libraries.
   Choose an appropriate one below for you system.

   hdf5-1.8.9-linux-static.tar.gz (for 32 bit systems)
   hdf5-1.8.9-linux-x86_64-static.tar.gz (for 64 bit systems)
   hdf5-1.8.9-linux-shared.tar.gz (for 32 bit systems)
```

hdf5-1.8.9-linux-x86_64-shared.tar.gz (for 64 bit systems)

a. These are precompiled binaries, so place them in /usr/local or any
   directory that you can source. Note these binaries are not compiled
   with parallel writing capability. You will need to compile from
   source for this capability.
   (This may be all you need to do with HDF5 directly.)

HDF5 SOURCE CODE to compile: You do not need to go this route if
the precompiled binaries work fine. Note that we had trouble compiling
parallel HDF5 on our 32bit system, but it worked fine on 64bit system.

hdf5-1.8.9.tar.gz (works for either 32 or 64 bit)

a. This is the uncompiled version. This allows you to complile on
   your own to generate libraries and executables. HDF5 ideally
   needs the zlib and szlib libraries. If these are not installed
   on you system, then install them before proceeding.
b1. First run with default compilers (something like):
   ./configure --prefix=/usr/local/hdf5-1.8.9
   Or run with specified compilers, such as:
   CC=gcc FC=gfortran ./configure --prefix=/usr/local/hdf5-1.8.9
b2. For parallel HDF5 compilation you will first need parallelization
   software installed such as MPICH or OPENMPI.
   For MPICH, try something similar to:
   ./configure --prefix=/usr/local/hdf5-1.8.9 \
   --disable-fortran --disable-fortran2003 \
   --enable-parallel CC=/usr/local/mpich2-1.4.1/bin/mpicc
c. Next do,
   make
   make check   # run test suite.
   make install   # you will need root permissions
   make check-install   # verify installation.

For either installation to run, you may need to update your environment
as examples show below:

a. In your .bash_profile, make sure the following are included.
   As an example:
   PATH=/usr/local/hdf5-1.8.9/bin
   LD_LIBRARY_PATH=/usr/local/hdf5-1.8.9/lib
b. The following MIGHT be necessary for access to HDF5 libraries:
   Create file '/etc/ld.so.conf.d/hdf5.conf' with 1 line that says
   for example:
   '/usr/local/hdf5-1.8.9/lib'

---

4. Third Party Compression Libararies may be necessary if you are doing a
   full build of HDF5. The precompiled versions come with libsz and libz.

   a. First you will need to install libsz and libz. These can be
      installed from packages szip-2.1.tar and zlib-1.2.5.tar. Automatic
      installation typically goes to a standard library location. Before
      installing these, see if they already exist on your Linux platform.
      If they do, then you should not need to perform this installation

```
   b. To install szip-2.1.tar, untar and enter directory, and proceed:
      You might need to be root user to install.
      ./configure --prefix=/usr  #this will install in /usr/lib
      make
      make check
      make install
   c. To install zlib-1.2.5, untar and enter directory, and proceed:
      You might need to be root use to install.
      prefix=/usr ./configure
        #Note: this will install in /usr/lib and /usr/include
      make
      make install

##########################################################################
##########################################################################
```

LINUX 32-BIT BASH-PROFILE SETUP FOR RAMS

```
##########################################################################
##########################################################################
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
      . ~/.bashrc
fi

# User specific environment and startup programs
PATH=/bin:/sbin:/lib
PATH=$PATH:/usr/bin:/usr/sbin:/usr/lib
PATH=$PATH:/usr/local/bin:/usr/local/sbin:/usr/local/lib

# INTEL COMPILER
PATH=$PATH:/opt/intel/composer_xe_2011_sp1.8.273/bin/ia32
# HDF5
PATH=$PATH:/usr/local/hdf5-1.8.9-linux-static/bin
# MPICH2
PATH=$PATH:/usr/local/mpich2-1.4.1/bin
# OPENMPI
PATH=$PATH:/usr/local/openmpi-1.6.5/bin
# GRIB TOOLS
PATH=$PATH:/home/smsaleeb/GribTools.2011/cnvgrib-1.1.9
PATH=$PATH:/home/smsaleeb/GribTools.2011/wgrib2.v1.8.2/wgrib2

PATH=$PATH:./

BASH_ENV=$HOME/.bashrc

#This first custom lib is for special necessary libraries
LD_LIBRARY_PATH=/home/smsaleeb/GribTools.2011/customlibs
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/mpich2-1.4.1/lib
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/openmpi-1.6.5/lib
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/hdf5-1.8.9-linux-static/lib
export LD_LIBRARY_PATH

export PGI=‘/opt/pgi’
export LM_LICENSE_FILE=‘$PGI/license.dat’
```

```
export BASH_ENV
unset USERNAME
export EDITOR=nano
export PATH

# Source the GEMPAK5.11.1 Gempak Environment File
export NAWIPS=/usr/local/GEMPAK5.11.1
if [ -f $NAWIPS/Gemenviron.profile ]; then
   . $NAWIPS/Gemenviron.profile
fi


##########################################################################
##########################################################################
```

LINUX 64-BIT BASH-PROFILE SETUP FOR RAMS

```
##########################################################################
##########################################################################
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
      . ~/.bashrc
fi

# User specific environment and startup programs
PATH=$PATH:$HOME/bin:$HOME/lib:/usr/bin:/usr/lib:/usr/lib64:/usr/include
PATH=$PATH:/usr/local/bin:/usr/local/lib:/lib:/lib64:/sbin

# INTEL COMPILER
PATH=$PATH:/opt/intel/composer_xe_2011_sp1.8.273/bin/intel64
# HDF5
PATH=$PATH:/usr/local/hdf5-1.8.9-linux-x86_64-static/bin
# MPICH2
PATH=$PATH:/usr/local/mpich2-1.4.1/bin
# OPENMPI
PATH=$PATH:/usr/local/openmpi-1.6.5/bin

PATH=$PATH:./

BASH_ENV=$HOME/.bashrc

LD_LIBRARY_PATH='/usr/lib64'
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/mpich2-1.4.1/lib
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/openmpi-1.6.5/lib
LD_LIBRARY_PATH=$LD_LIBRARY_PATH: \
    /usr/local/hdf5-1.8.9-linux-x86_64-static/lib

export LD_LIBRARY_PATH

#export PGI='/usr/local/pgi'
#export LM_LICENSE_FILE='$PGI/license.dat'

export BASH_ENV
```

```
unset USERNAME
export EDITOR=nano
export PATH

# Source the GEMPAK5.11.1 Gempak Environment File
export NAWIPS=/usr/local/GEMPAK5.11.1
if [ -f $NAWIPS/Gemenviron.profile ]; then
   . $NAWIPS/Gemenviron.profile
fi
```