[circa 2014-12-09; PRAMS_2.9]

Obtain and install the necessary files

Prerequisites:
1) Fortran (95+) compiler (*e.g.*, gfortran 4.6+, ifort, pgf90)
2) C compiler (*e.g.*, gcc, icc, pgcc)
3) Python 2.7+
4) NetCDF (must have been compiled with Fortran 90+ bindings; version 4+ is preferable; http://www.unidata.ucar.edu/software/netcdf/)

Optional Prerequisites:
5) MPI [*e.g.*, Open MPI (http://www.open-mpi.org) or MPICH2 (http://www.mcs.anl.gov/research/projects/mpich2/); must have been compiled with Fortran 90+ bindings] for parallel runs
6) NCL (http://www.ncl.ucar.edu/) for grid placement visualization

Install and configure the universal_lib source code tree:

From a compressed archive: Decompress and extract the universal_lib archive in a directory of your choice (*e.g.*, */home/user/PRAMS* ; will automatically be unpacked into a subdirectory named *universal_lib)*:

```
bzip2 -dc universal_lib-1.2_r10-fs_dist.tar.bz2 | tar xvf - ;
```

-OR-

From the bitbucket.org repository using git: Change to a directory of your choice (*e.g.*, */home/user/PRAMS*) and execute the below command (the placeholder *bitbucket_repository_URL* should be something like `https://some_user@bitbucket.org/some_user/universal_lib.git`). The repository contents will automatically be placed into a new subdirectory named *universal_lib*:

```
git clone {bitbucket_repository_URL};
```

Change directory:

```
cd universal_lib/infrastructure/build/build_env_config;
```

Copy *user_change_me-inclibs* **and** the "*user_change_me-**" files most relevant to your computer system to this directory – for example:

```
cp examples/user_change_me-inclibs . ;
cp examples/gfortran_gcc-linux/* . ;
```

Edit the "user_change_me-*" files as needed (*e.g.*, with specific compiler options), testing the success of the compilation via the following (iteration may be needed, along with inspection of the on-screen output and *../configure_build_env/work/config.log*):

```
../../../admin_script.py clean ALL; ../../../admin_script.py build;
```

NOTE: Only the NETCDF_* and MPI_* entries in *user_change_me-inclibs* need to be correct (for PRAMS, the NCL_NCARg_* and CFITSIO_* entries can be ignored).

If you encounter a compilation error involving something not found in module *mpi*, try adding "`-DBROKEN_MPI_MOD`" to your universal_lib *user_change_me-compilers.** files (then clean, and compile again).

Install and configure the PRAMS source code tree:

From compressed archives: Decompress and extract the PRAMS code archives in a directory of your choice (*e.g.*, */home/user/PRAMS*; will automatically be unpacked into subdirectories named *common* and *Mars)*:

```
bzip2 -dc PRAMS_common-2.9_r9-fs_dist.tar.bz2 | tar xvf - ;
bzip2 -dc PRAMS_Mars-2.9_r9-fs_dist.tar.bz2 | tar xvf - ;
```

-OR-

From the bitbucket.org repository using git: Change to a directory of your choice (*e.g.*, */home/user/PRAMS*) and execute the below (the placeholder *bitbucket_repository_URL_for_** should be something like `https://some_user@bitbucket.org/some_user/PRAMS.git`). The contents of the repository will automatically be placed into a new subdirectory named *PRAMS*:

```
git clone {bitbucket_repository_URL_for_PRAMS};
```

Change directory:

```
cd Mars/infrastructure/build;
```

Make a copy of *build_env_config.other_packages-template* called *build_env_config.other_packages*, then edit the new file appropriately (to specify where the relevant *universal_lib* and *common* directories are located):

```
cp build_env_config.other_packages-template
     build_env_config.other_packages;
```

Change directory:

```
cd build_env_config;
```

Copy the "user_change_me-*" files most relevant to your computer system to this directory – for example:

```
cp examples/gfortran_gcc-linux/* . ;
```

Edit the "*user_change_me-**" files as needed (*e.g.*, with specific compiler options), testing the success of the compilation via the following (iteration may be needed, along with inspection of the on-screen output and *../../../../common/infrastructure/build/configure_build_env/work/config.log*):

```
../../../admin_script.py clean ALL; ../../../admin_script.py build;
```

Build the modeling system

Change directory to *Mars*.

List the possible options available:

```
./admin_script.py -h;
```

Typically, one would build the modeling system with the following commands:

```
./admin_script.py build;     ( serial)
./admin_script.py build DM_only;     ( parallel)
```
-OR-
```
./admin_script.py build debug;     (serial, with debugging flags)
./admin_script.py build DM_only debug;     (parallel, with debugging flags)
```

Install the desired static data files

This step does not necessarily have to be done every time – it is likely that one would not want too many copies of this on a single machine/filesystem, as these files (in total) are several GiB in size.

Decompress and extract the PRAMS data archives in a directory of your choice (*e.g.*, */data/user/input_static-PRAMS_2.9*; will automatically be unpacked into subdirectories named *common* and *Mars)*:

```
bzip2 -dc PRAMS_2.9-v1.common.full_data.tar.bz2 | tar xvf - ;
bzip2 -dc PRAMS_2.9-v3.Mars.smaller_data.tar.bz2 | tar xvf - ;
```
(**OPTIONAL**):
```
bzip2 -dc PRAMS_2.9-v1.Mars.large_data.tar.bz2 | tar xvf -;
```

Prepare the run directory

This version of PRAMS offers a significant amount of flexibility regarding where its input and output data are located. However, in order to easily refer to those locations, it is suggested that a set of symbolic links pointing to those locations be created in the *run* directory. Also, in choosing a location for the PRAMS output, bear in mind that typical model output from a single PRAMS simulation can range in size from < 10 GiB to > 100 GiB, so ensure that the chosen directory resides on a data volume that can store significant quantities of data.

Change directory to *Mars/run*

Examples of creating such symbolic links:
`ln -s` *{dir_where_the_static_data_files_are}* `input_static;`
`ln -s` *{dir_where_the_GCM_output_data _are}* `MGCM_output;`
`ln -s` *{dir_for_PRAMS_output}* `output;`

```
cp run_PRAMS-template run_PRAMS;
cp run_postp-template run_postp;
```

Running the model

Prepare a model configuration/namelist file (*e.g.*, *PRAMS_IN.test*; use PRAMS_IN-template as a template). The general way to run the model (in serial) is:

```
./run_PRAMS -f PRAMS_IN.test;
```

For a simulation with INITIALIZATION_TYPE = 2:
1) Set RUN_TYPE = 'MAKE_VAR_FILES' in the namelist, and run the model.
2) Then set RUN_TYPE = 'INITIAL' in the namelist, and run the model.

For a simulation with INITIALIZATION_TYPE = 1:
1) Set RUN_TYPE = 'INITIAL' in the namelist, and run the model.

To run in parallel with the computational load split between 6 nodes, with one supervisory/root node (note that the model must be compiled for parallel for this to work):

```
./run_PRAMS -n 7 -f PRAMS_IN.test;
```

Updating the codebase(s)

With "official" compressed archive images: To update your codebase with an "official" archive image that you have obtained, use the install mode of the appropriate *admin_script.py* – note that the **.tar.bz2* can be in any directory, and will not be deleted or changed. Be aware that any locally-modified source code with the same names will be overwritten. Some examples:

```
cd PRAMS/PRAMS/common;
./admin_script.py install PRAMS_common-2.9_r10-fs_dist.tar.bz2;

cd PRAMS/PRAMS/Mars;
./admin_script.py install PRAMS_Mars-2.9_r10-fs_dist.tar.bz2;

cd PRAMS/universal_lib;
./admin_script.py install universal_lib-1.2_r11-fs_dist.tar.bz2;
```

-OR-

With the bitbucket.org repository using git: A git-aware repository/directory (*i.e.*, `git status` doesn't return an error) already contains the bitbucket.org URL information, and can be updated as in the following examples (if you have any locally-modified source code, git may complain and suggest alternative courses of action – but that is beyond the scope of this guide):

```
cd PRAMS/PRAMS/common;
git pull;

cd PRAMS/PRAMS/Mars;
git pull;

cd PRAMS/universal_lib;
git pull;
```