Wikiprint Book

Title: WRF4G2.0/Preprocessor

Subject: TracMeteo - WRF4G2.0/Preprocessor

Version: 9

Date: 08/19/2022 10:42:15 PM

Table of Contents

Why a preprocessor?

Before the WRF Preprocessor System (WPS) runs, input data files need to be in a specific format, so `ungrib.exe` is able to parse them. Specifically, files in ?GRIB format are needed. Furthermore, to be able to run WRF, there are some mandatory fields that must be presented in the input files. Some of these are not always available in the driving model data we want to use, such as soil temperature and moisture content. In addition, it is needed these variables with the frequency that WRF is going to use to update its boundary data (usually 6 hours).

Thus, in many cases input data available in some online server need to be preprocessed. This process could be carried out offline, transforming the whole downloaded dataset before using it. However, integrating it in the WRF4G workflow enables the user to download and preprocess the input data locally in any of the grid nodes. In WRF4G the program or script that performs this process is called `preprocessor`.

preprocessors in WRF4G

As mentioned, the preprocessor is called before the WPS in each chunk. It must be an executable file that can be written in any language, and accepts 5 arguments. These are the `initial date`, the `end date` (as YYYY-mm-dd HH:MM:SS), the `input data path`, `member number` and `initialization month number` (4 and 5 arguments are available when `extdata_member` variable is used). After preprocessing, the preprocessor must copy the properly formatted input data to a folder called `WPS/grbdata`.

WRF4G provides some preprocessors as examples. These are located under `$WRF4G_DEPLOYMENT_DIR/repository/apps/preprocessor` directory. In order to create preprocessors, they have to be located in the `wrf4g_files/bin` directory of the specific experiment.

Sample preprocessor:

```
#! /bin/bash
#
# Sample preprocessor in shell.

sformateddate=$1     # initial date to process as YYYY-MM-DD_HH:MM:SS (Ex. 1983-08-27_00:00:00)
eformateddate=$2     # end date to process
extdata_path=$3      # Path to the data
#
# and creating a directory grbData with grib data for those dates and ready to
# be linked by 'link_grib.csh grbData/*.grb'
#
read iyy imm trash <<< `echo $sformateddate | tr '_T:-' '    '`
read fyy fmm trash <<< `echo $eformateddate | tr '_T:-' '    '`

function get_yearmons(){
 yeari=$1
 moni=$2
 yearf=$3
 monf=$4
 yearmoni="$yeari$(echo $moni | awk '{printf "%02d",$1}')"
 yearmonf="$yearf$(echo $monf | awk '{printf "%02d",$1}')"
 for year in $(seq $yeari $yearf); do
   for month in $(seq 1 12); do
     thisyearmon="$year$(printf "%02d" $month)"
     if test $thisyearmon -ge $yearmoni -a $thisyearmon -le $yearmonf; then
       echo ${thisyearmon}
     fi
   done
 done
}

echo "Linking global data from: ${extdata_path}"
mkdir -p grbData
for yearmon in $(get_yearmons $iyy $imm $fyy $fmm)
do
    year=${yearmon:0:4}
    wrf4g vcp ${extdata_path}/${year}/'*'${yearmon}'*'.grb ln://`pwd`/grbData/
done
```