GEFS in Subversion

Dick Wobus

November 10, 2010

What does Subversion do?

- Keep track of the development of a version of GEFS (scripts, codes, parms)
- Keep a complete history of changes, so that any earlier version can be retrieved
- Keep a “trunk” version, which in our case will emulate as closely as possible the current production version, and as many “branch” versions as we want to define
- Enable us to identify differences between branches
- Enable us to merge changes from one branch to another
- Manage code and scripts in the manner that EMC and NCO are adopting as standard

How have we managed versions for the last several years?

- Each version of the parallel GEFS that I create and maintain is in a directory tree /global/save/wx20rw/s/rwLL/nwdev, where LL is a two letter version identifier (latest version is rwtt) which contains subdirectories jobs, scripts, sorc, etc. corresponding to those under /nwprod, but containing only GEFS scripts, codes, etc.
- Additional subdirectory control contains our control scripts, which manage jobs in place of SMS, and an sms subdirectory which contains examples of the production SMS scripts which we are emulating
- When I want to start a new version, to add a feature or to merge in features from some other version, I copy into a new subdirectory with new identifier rwLL
- You can use these versions by copying into subdirectories /global/save/$LOGNAME/…./IDLL/nwdev where ID is your two-letter unique identifier, for example, your initials, and LL is your two-letter version identifier – IDLL has to contain 4 characters because parts of it are used to create output subdirectory names and job identifiers.
- I keep /global/save/wx20rw/s/doc.txt which contains one- or two-line descriptions of each version, including the name of the version that it is primarily based on

- Recent experiments

- new:
- grep "rwLL__" to select the description of one experiment
- grep "rwLL" to select all references to one experiment

- old:
- grep "nwgeLL" to select the description of one experiment
- grep "geLL" to select all references to one experiment

- rwrt__ rwrq restrict large jobs to certain hours

- rwrs__ rwrp change cp to mv for post restart copies

- rwrr__ rwrp add RFC changes
- rwrr__ rwrp current candidate for ops+fix branch

- rwrq__ rwro modify fhmax and fhmaxh logic, use new fcst code

- rwrp__ rwrn modify fhmax and fhmaxh logic, use prod fcst code
- rwrp__ rwrn current candidate for subversion trunk

- rwro__ rwrj test Dingchen's system for ensstat problem

- rwrn__ rwrg clean up gefs.parm for subversion

- rwrm__ rwrk RFC system to fix anomcat problem

- rwrl__ rwri modify to analyze anomcat problem

- rwrk__ rwri RFC system for TIGGE II and bug fix

- rwrj__ rwrg with Dingchen's mods for new fcst code

- rwri__ rwrh with rwre code and other RFC changes

- rwrh__ rwrg newly compiled forecast code

- rwrg__ rwrc update to 201007 implementation

- rwrf__ rwrc control for rwrd and rwre

- rwre__ rwrd TIGGE II + bug fixes

Latest versions in doc.txt

Replacing this process with subversion

- Our process has been modified in several ways over the past 2 years so as to be more easily managed using subversion – for example, the “environment” variable is now dev instead of the version identifier, so that job script names do not have to change for each version
- My linear list of versions will be superseded by a single trunk version (corresponding to current production) and a series of branch versions corresponding to upcoming implementations and other versions I maintain for the group to use. The trunk and branch structure will make it easier to keep track of the “family tree” of development
- Any version(s) you develop will be in your own branch(es)
- To run a version (current or past) you will check out a copy from subversion into a subdirectory with a 4-letter identifier name (similar to what you would copy into now, the control scripts require the 4-letter identifier)
- After you make changes that you want to keep track of, you can commit this version to your branch
- When you or I commit an updated version, we enter a description of the change into the log, which corresponds to the one-line entries in doc.txt
- When your changes are ready to be included in a production version (or in a version to be tested for implementation) we will merge them into one of my branches or into the trunk

Branches that I am preparing to create

- Fix copy for mirror
- Fix copy and bugs + TIGGE II
- Fixes + TIGGE II + option to run current GFS code
- Fixes + TIGGE II + option to run current GFS code + option to reduce resolution after 192 hr

Resources for using subversion

- Online command help: svn help
- “cheat sheet” by Paul van Delst http://www2.emc.ncep.noaa.gov/documents/EMC.svn_intro.cheatsheet.pdf (on EMC IT trouble desk menu under documents)
- Account application, on the same web page
- Subversion forums http://optimus.ncep.noaa.gov/forum/
- Seminars given by Paul van Delst