Testing NCEPLIBS-bufr

Jeff Ator

NCEPLIBS-bufr

- Software library for encoding and decoding BUFR, which is a standard WMO format for worldwide observational data exchange
- Available from https://github.com/NOAA-EMC/NCEPLIBS-bufr
  - Latest release version is 12.0.1, with next 12.1.0 due out soon
- Used by numerous application codes inside and outside of NCEP
- Original version of library dates back to early 1990's
- Now roughly 10K lines of code across nearly 350 subprograms
- Yet now with nearly 96% testing coverage!
- So, how did we get there?

GCC Code Coverage Report

Directory: ./
Date: 2024-03-30 00:25:20
Coverage: low: ≥ 0% medium: ≥ 75.0% high: ≥ 90.0%

	Exec	Total	Coverage
Lines:	9489	9940	95.5%
Functions:	347	347	100.0%
Branches:	5749	7898	72.8%
Decisions:	850	966	88.0%

Getting started with cmake

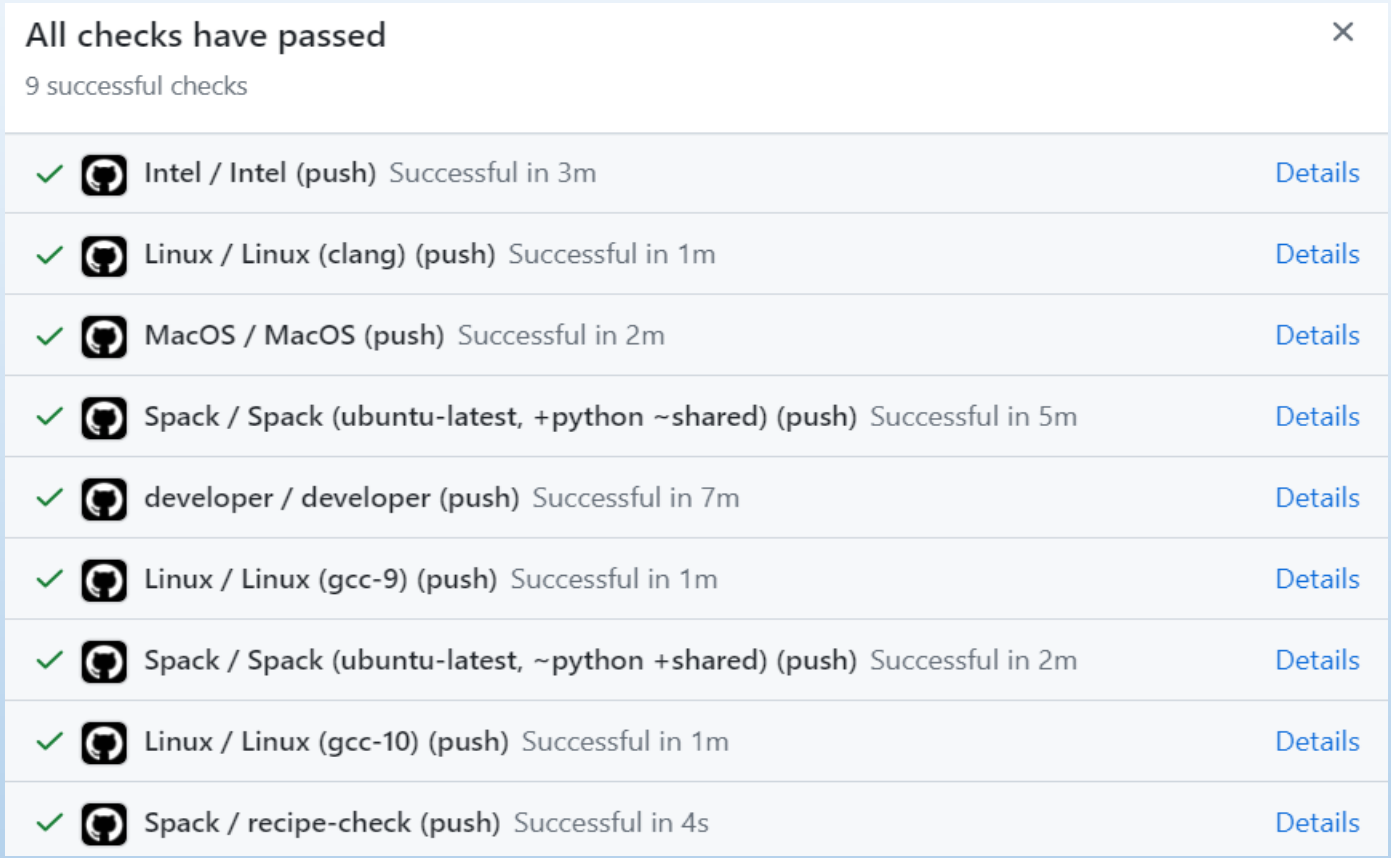
- Create a new “test” subdirectory in the project to contain your test codes
- In your CMakeLists.txt file, build your test codes and use `add_test()` to include each one in your automated CI test suite
- In your .github/workflows directory, add a new “test” block sequence to your “steps:” mapping within the YAML file for each supported platform
  - Use `ctest` command to execute your entire suite of test codes on that platform, including recommended options `--verbose --output-on-failure --rerun-failed`
  - Any test returning with a non-zero value is considered a fail on that platform, which in turn prevents any merging to the baseline whenever the test is included as part of each PR
  - Recommend to include at least one YAML build for GNU with all debug and compiler warnings enabled, including `-Werror -Wall --fsanitize=address`
  - If also include `-ftest-coverage` compiler flag, then can measure overall test coverage with `gcovr -v --html-details test-coverage.html` command in the “test” block sequence, immediately following the above `ctest` command

```
      Start  1: run_test_bort
 1/90 Test  #1: run_test_bort ...................   Passed  229.33 sec
      Start  2: test_c_misc
 2/90 Test  #2: test_c_misc .....................   Passed    0.15 sec
      Start  3: test_c_cread
 3/90 Test  #3: test_c_cread ....................   Passed    0.16 sec
      Start  4: test_c_crwbmg
 4/90 Test  #4: test_c_crwbmg ...................   Passed    0.14 sec
      Start  5: test_misc_4
 5/90 Test  #5: test_misc_4 .....................   Passed    1.10 sec
      Start  6: test_ufbrw_4
 6/90 Test  #6: test_ufbrw_4 ....................   Passed    3.20 sec
      Start  7: test_ufbcup_4
 7/90 Test  #7: test_ufbcup_4 ...................   Passed    0.64 sec
      Start  8: intest1_4
 8/90 Test  #8: intest1_4 .......................   Passed    0.45 sec
      Start  9: intest2_4
 9/90 Test  #9: intest2_4 .......................   Passed    0.40 sec
      Start 10: intest3_4
10/90 Test #10: intest3_4 .......................   Passed    0.76 sec
               . . .
      Start 16: intest9_4
16/90 Test #16: intest9_4 .......................   Passed    0.43 sec
      Start 17: intest10_4
17/90 Test #17: intest10_4 ......................   Passed    0.31 sec
      Start 18: intest11_4
18/90 Test #18: intest11_4 ......................   Passed    0.38 sec
      Start 19: intest12_4
19/90 Test #19: intest12_4 ......................   Passed    0.54 sec
      Start 20: outtest1_4
20/90 Test #20: outtest1_4 ......................   Passed    0.34 sec
      Start 21: outtest2_4
21/90 Test #21: outtest2_4 ......................   Passed    0.60 sec
      Start 22: outtest3_4
22/90 Test #22: outtest3_4 ......................   Passed    0.36 sec
               . . .
      Start 28: outtest9_4
28/90 Test #28: outtest9_4 ......................   Passed    0.36 sec
      Start 29: outtest10_4
29/90 Test #29: outtest10_4 .....................   Passed    0.98 sec
               . . .
      Start 80: test_c_interface_4
80/90 Test #80: test_c_interface_4 ..............   Passed    1.12 sec
      Start 81: test_c_interface_2_4
81/90 Test #81: test_c_interface_2_4 ............   Passed    0.40 sec
      Start 82: test_debufr
82/90 Test #82: test_debufr .....................   Passed   23.46 sec
      Start 83: test_gettab
83/90 Test #83: test_gettab .....................   Passed    0.70 sec
      Start 84: test_sinv
84/90 Test #84: test_sinv .......................   Passed   11.24 sec
      Start 85: test_binv
85/90 Test #85: test_binv .......................   Passed    1.29 sec
      Start 86: test_split_by_subset
86/90 Test #86: test_split_by_subset ............   Passed    2.75 sec
      Start 87: test_readbp
87/90 Test #87: test_readbp .....................   Passed   24.53 sec
      Start 88: test_readmp
88/90 Test #88: test_readmp .....................   Passed    0.94 sec
      Start 89: test_cmpbqm
89/90 Test #89: test_cmpbqm .....................   Passed   37.33 sec
      Start 90: test_xbfmg
90/90 Test #90: test_xbfmg ......................   Passed    1.24 sec

100% tests passed, 0 tests failed out of 90

Total Test time (real) = 389.35 sec
```

ctest output



As part of each PR, we currently run 90 tests
(including various _4, _8, _d build configurations),
on each of 9 different platforms!

gcovr output

GCC Code Coverage Report

Directory: ./
Date: 2024-03-30 00:25:20
Coverage: low: ≥ 0% medium: ≥ 75.0% high: ≥ 90.0%

	Exec	Total	Coverage
Lines:	9489	9940	95.5%
Functions:	347	347	100.0%
Branches:	5749	7898	72.8%
Decisions:	850	966	88.0%

List of functions

File	Lines	Functions	Branches	Decisions
build/src/bfrini.F90	100.0% 60 / 60	100.0% 1 / 1	100.0% 22 / 22	100.0% 2 / 2
build/src/bvers.f	100.0% 6 / 6	100.0% 1 / 1	66.7% 4 / 6	-% 0 / 0
build/utils/debufr.c	88.7% 141 / 159	100.0% 2 / 2	75.0% 33 / 44	78.0% 32 / 41
build/utils/sinv.F90	96.2% 50 / 52	100.0% 3 / 3	73.3% 22 / 30	100.0% 2 / 2
src/arallocc.c	100.0% 40 / 40	100.0% 2 / 2	50.0% 15 / 30	50.0% 15 / 30
src/arallocf.F90	100.0% 581 / 581	100.0% 5 / 5	52.3% 1396 / 2670	100.0% 2 / 2
src/blocks.F90	95.7% 22 / 23	100.0% 2 / 2	83.3% 10 / 12	83.3% 5 / 6
src/borts.F90	100.0% 17 / 17	100.0% 2 / 2	-% 0 / 0	-% 0 / 0
src/bufr_c2f_interface.F90	99.6% 227 / 228	100.0% 52 / 52	57.2% 103 / 180	58.3% 14 / 24
src/capit.f	100.0% 10 / 10	100.0% 1 / 1	100.0% 6 / 6	-% 0 / 0
src/cfe.c	98.9% 86 / 87	100.0% 7 / 7	89.7% 52 / 58	96.4% 27 / 28
src/chekstab.f	77.8% 14 / 18	100.0% 1 / 1	78.6% 11 / 14	-% 0 / 0
src/cidecode.F90	100.0% 73 / 73	100.0% 7 / 7	84.2% 32 / 38	100.0% 14 / 14
src/ciencode.F90	100.0% 66 / 66	100.0% 4 / 4	74.4% 58 / 78	93.8% 15 / 16
src/cktaba.f	94.0% 78 / 83	100.0% 1 / 1	86.1% 31 / 36	-% 0 / 0
src/closmg.f	100.0% 23 / 23	100.0% 1 / 1	93.8% 15 / 16	-% 0 / 0
src/cmpmsg.f	100.0% 7 / 7	100.0% 1 / 1	100.0% 2 / 2	-% 0 / 0
src/cmsgini.f	94.2% 49 / 52	100.0% 1 / 1	50.0% 4 / 8	-% 0 / 0
src/cnved4.f	98.0% 48 / 49	100.0% 1 / 1	80.0% 8 / 10	-% 0 / 0
src/codflg.f	100.0% 7 / 7	100.0% 1 / 1	100.0% 2 / 2	-% 0 / 0
src/conwin.f	100.0% 11 / 11	100.0% 1 / 1	100.0% 8 / 8	-% 0 / 0
src/copydata.F90	97.3% 214 / 220	100.0% 9 / 9	89.3% 109 / 122	88.4% 99 / 112
src/cpdxmm.f	90.7% 39 / 43	100.0% 1 / 1	86.4% 19 / 22	-% 0 / 0
src/cpmstabs.c	100.0% 27 / 27	100.0% 1 / 1	100.0% 20 / 20	100.0% 21 / 21
src/cread.c	96.7% 58 / 60	100.0% 8 / 8	95.0% 19 / 20	95.0% 19 / 20
src/crwbmg.c	100.0% 60 / 60	100.0% 5 / 5	88.1% 37 / 42	88.6% 31 / 35
src/drfini.f	91.3% 21 / 23	100.0% 1 / 1	87.5% 7 / 8	100.0% 1 / 1
src/drstpl.f	100.0% 14 / 14	100.0% 1 / 1	87.5% 7 / 8	-% 0 / 0

141				
142			3792	call status(lunot,lot,il,im)
143	▶ 2/2	▶ 2/2	3792	if(il.eq.0) call bort('BUFRLIB: COPYMG - OUTPUT BUFR FILE IS CLOSED, IT MUST BE OPEN FOR OUTPUT')
144	▶ 2/2	▶ 2/2	3790	if(il.lt.0) call bort('BUFRLIB: COPYMG - OUTPUT BUFR FILE IS OPEN FOR INPUT, IT MUST BE OPEN FOR OUTPUT')
145	▶ 1/2	▶ 1/2	3788	if(im.ne.0) call bort('BUFRLIB: COPYMG - ALL MESSAGES MUST BE CLOSED IN OUTPUT BUFR FILE, A MESSAGE IS OPEN')
146				
147				! Make sure both files have the same tables
148				
149			3788	subset = tag(inode(lin))(1:8)
150			3788	call nemtba(lot,subset,mtyp,msbt,inod)
151	▶ 1/2	▶ 1/2	3788	if(inode(lin).ne.inod .and. iok2cpy(lin,lot).ne.1) &
152			✗	call bort('BUFRLIB: COPYMG - INPUT AND OUTPUT BUFR FILES MUST HAVE THE SAME INTERNAL TABLES, THEY ARE DIFFERENT HERE')
153				
154				! Everything okay, so copy a message
155				
156			3788	mbym = iupbs01(mbay(1,lin),'LENM')
157			3788	call msgwrt(lunot,mbay(1,lin),mbym)
158				
159				! Set the message control words for partition associated with lunot
160				
161			3786	nmsg(lot) = nmsg(lot) + 1
162			3786	nsub(lot) = msub(lin)
163			3786	msub(lot) = msub(lin)
164			3786	idate(lot) = idate(lin)
165			3786	inode(lot) = inod
166				
167			3786	return
168				end subroutine copymg

Allows you to quickly and easily see which parts of your package need more testing, so you can write new tests to focus on those areas!

Summary

- Setting up automated CI testing does take some time and effort
  - Especially for large legacy codes which never previously had it
- But the payoff is you end up with a code package you can have much greater confidence in
  - With minimal ongoing maintenance
  - Easier to support across numerous platforms
  - Add new library features without worrying about breaking existing functionality
  - Always add new tests whenever you add new code or functionality
  - Never remove existing tests unless the underlying code is also removed from the library