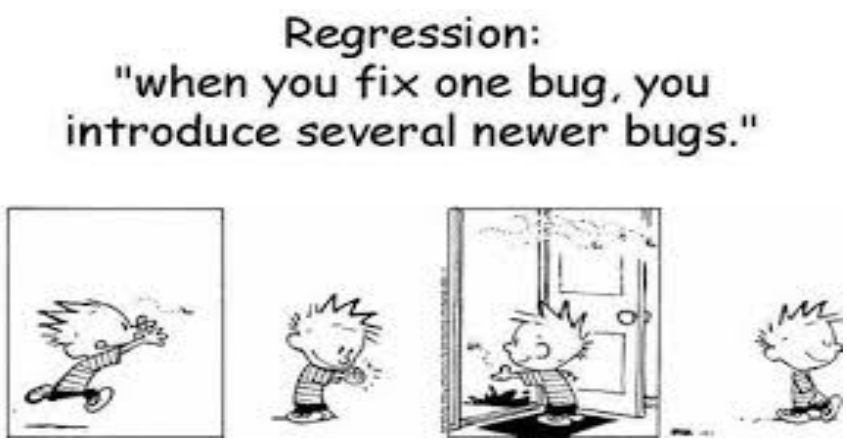
Waves2Weather — ECS Workshop

Z2, Scientific Programmers at Your Service

LMU — Meteorological Institute Munich



Jan 21, 2019

Developing scientific code

- write function
- print result and check for correctness
- check use cases, edge cases, invalid input

Developing scientific code

- write function
- print result and check for correctness
- check use cases, edge cases, invalid input

---

adding tests to it . . .

- record output and form automated tests
- check for physical constraints (i.e. analytic solutions, symmetries, conservation properties)
- if you find a bug, make a test out of it

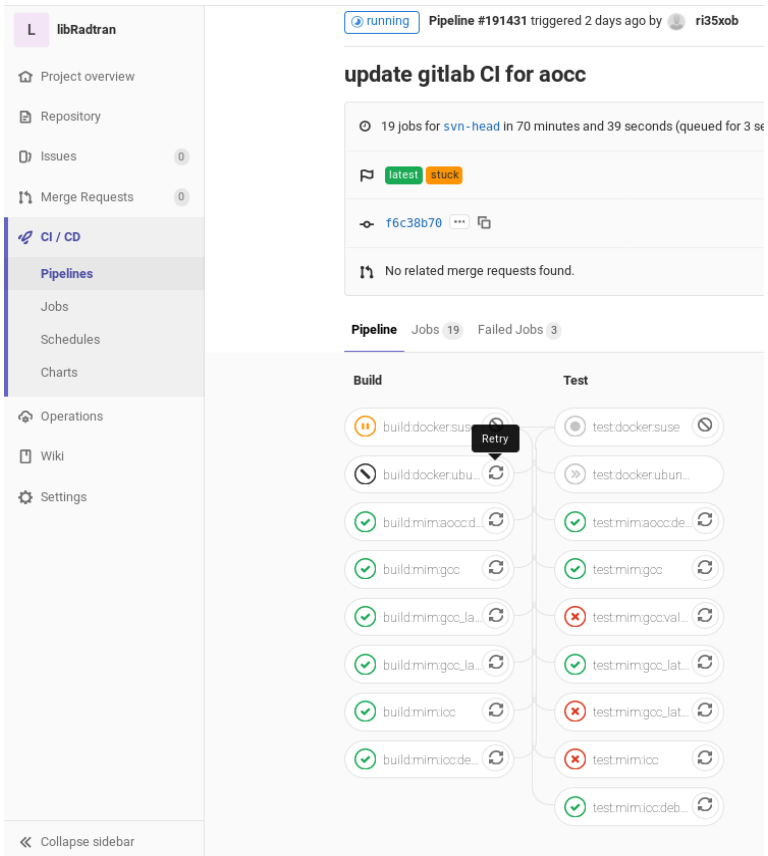
Runtime Assertions (defensive programming)

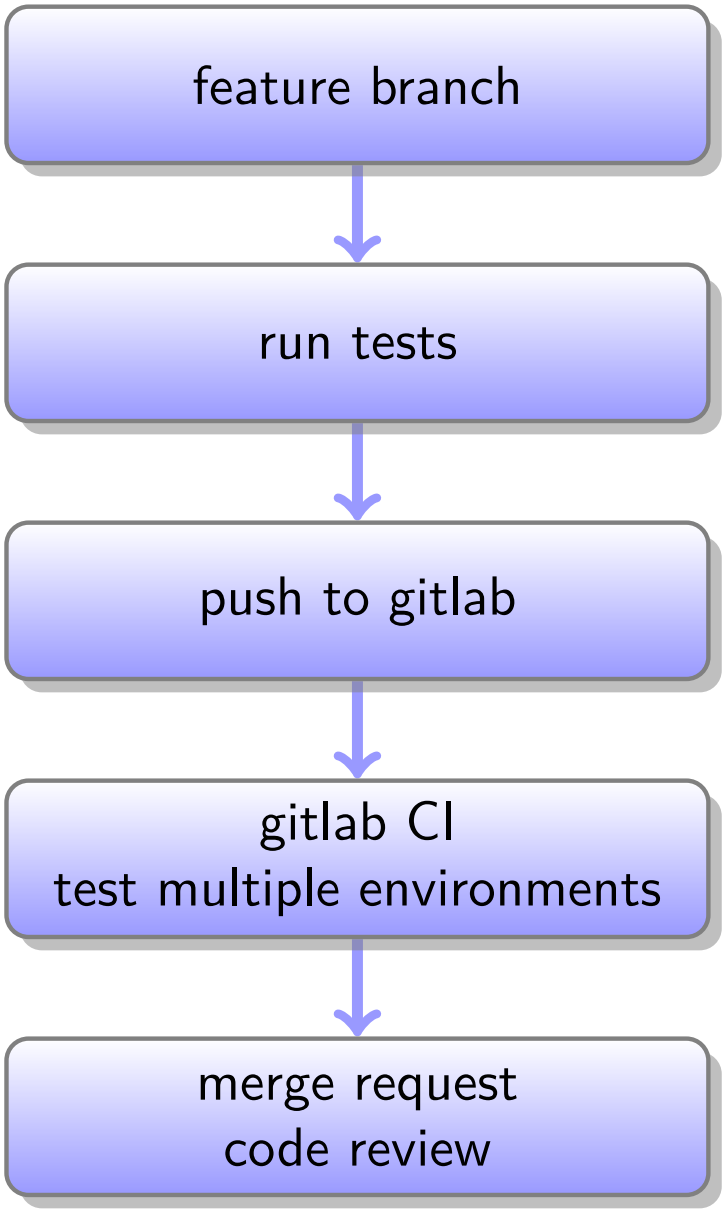
- check status of input
- check intermediate values
- immediate exit simplifies debugging
- always return status and check it! $\rightarrow$ traceback

Testing buzzword bingo

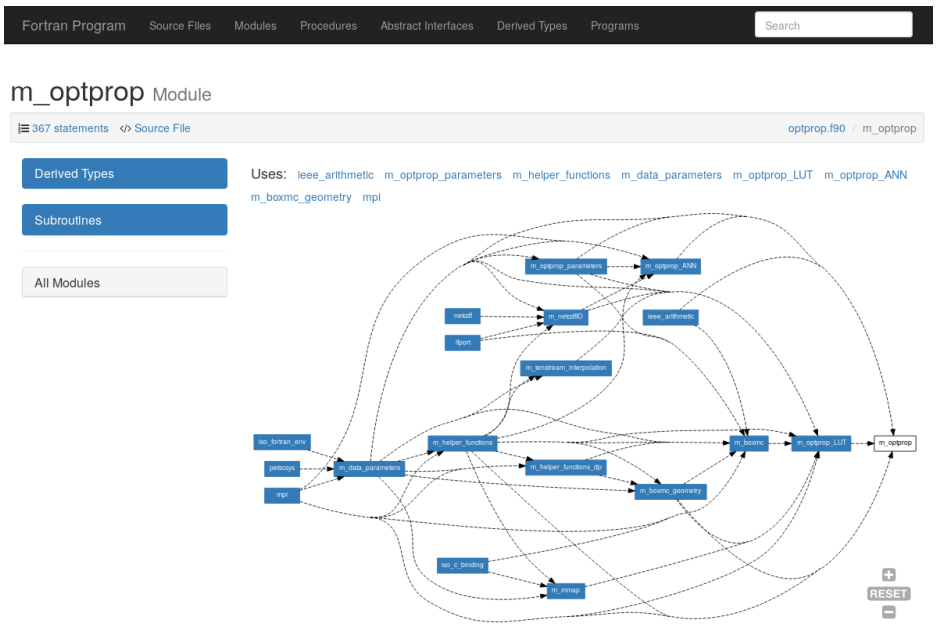
- unit tests — single function
- integration tests — combination of parts
- regression tests — behaviour stays the same
- full system tests — test full model
- test driven development
- automated testing & continuous integration

Running tests automatically, Continous Integration





How to work with large codebases?



- doxygen — generates automated documentation

How to work with large codebases?

```
-DHAVE_SOS=1 -DHAVE_LIBGSL=1   -DLRTVERSION="2.0.2-MYSTIC"   molecular3d.c
molecular3d.c: In function 'read_atmosphere_3d':
molecular3d.c:584:10: warning: unused variable 'N' [-Wunused-variable]
   size_t N;
          ^
molecular3d.c: In function 'write_dummy_profile_file':
molecular3d.c:1010:12: warning: unused variable 'tmpdata' [-Wunused-variable]
   float ***tmpdata=NULL;
            ^~~~~~
molecular3d.c:1008:41: warning: unused variable 'id_qH2O' [-Wunused-variable]
   int    id_z=0, id_press=0, id_temp=0, id_qH2O=0;
                                         ^~~~~~
molecular3d.c:1008:30: warning: unused variable 'id_temp' [-Wunused-variable]
   int    id_z=0, id_press=0, id_temp=0, id_qH2O=0;
                              ^~~~~~
molecular3d.c:1008:18: warning: unused variable 'id_press' [-Wunused-variable]
   int    id_z=0, id_press=0, id_temp=0, id_qH2O=0;
                  ^~~~~~~~
molecular3d.c:1002:10: warning: unused variable 'N' [-Wunused-variable]
   size_t N;
          ^
molecular3d.c:1001:18: warning: unused variable 'iystart' [-Wunused-variable]
   int ixstart=0, iystart=0;
                  ^~~~~~~
molecular3d.c:1001:7: warning: unused variable 'ixstart' [-Wunused-variable]
   int ixstart=0, iystart=0;
       ^~~~~~~
molecular3d.c:984:38: warning: unused variable 'iy' [-Wunused-variable]
   int i=0, number=0, status=0, ix=0, iy=0, iz=0, lc=0;
                                      ^~
molecular3d.c:984:32: warning: unused variable 'ix' [-Wunused-variable]
   int i=0, number=0, status=0, ix=0, iy=0, iz=0, lc=0;
                                ^~
```

- doxygen — generates automated documentation
- various compilers with warnings (gcc, icc, nag, aocc)
- learn to use debuggers (e.g. gdb, ipdb)

How to work with large codebases?

Bug Summary

Bug Type	Quantity	Display?
All Bugs	**1291**	☑
API		
Argument with 'nonnull' attribute passed null	15	☑
Dead store		
Dead assignment	288	☑
Dead increment	4	☑
Logic error		
Assigned value is garbage or undefined	3	☑
Dangerous variable-length array (VLA) declaration	4	☑
Dereference of null pointer	138	☑
Division by zero	1	☑
Garbage return value	1	☑
Result of operation is garbage or undefined	8	☑
Uninitialized argument value	20	☑
Memory error		
Memory leak	700	☑
Use of zero allocated	16	☑
Use-after-free	3	☑
Unix API		
Allocator sizeof operand mismatch	90	☑

- doxygen — generates automated documentation
- various compilers with warnings (gcc, icc, nag, aocc)
- learn to use debuggers (e.g. gdb, ipdb)
- static code analyzers (e.g. clang scan-build)

How to work with large codebases?

Allocator sizeof operand mismatch	src/spline.c	main	315	1	View Report
Allocator sizeof operand mismatch	src/ancillary.c	get_number_from_netCDF_map	9495	1	View Report
Allocator sizeof operand mismatch	src/ancillary.c	get_number_from_netCDF_map	9509	1	View Report
Allocator sizeof operand mismatch	libsrc_c/gencld.c	calc_new_profile	382	1	View Report
Allocator sizeof operand mismatch	libsrc_c/nca.c	nca3d_3	1344	1	View Report
Allocator sizeof operand mismatch	libsrc_c/netCDF_functions.c	write_netCDF_3D_byte	840	1	View Report
Allocator sizeof operand mismatch	src/cloud.c	read_ECMWF_clouds	5016	1	View Report
Allocator sizeof operand mismatch	src/ck.c	crs_ck	1815	1	View Report
Allocator sizeof operand mismatch	libsrc_c/lidar.c	summarize_result_lidar	4087	1	View Report
Allocator sizeof operand mismatch	libsrc_c/mystic.c	calloc_1D_atmos	19398	1	View Report
Allocator sizeof operand mismatch	src/mie.c	mie	361	1	View Report
Allocator sizeof operand mismatch	libsrc_c/netCDF_functions.c	read_3d_char	1574	1	View Report
Allocator sizeof operand mismatch	libsrc_c/nca.c	nca3d_3	1410	1	View Report
Allocator sizeof operand mismatch	src/uvspec_lex.c	calluvspec	17658	1	View Report
Allocator sizeof operand mismatch	libsrc_c/nca.c	nca3d_3	1281	1	View Report
Allocator sizeof operand mismatch	libsrc_c/mystic.c	calloc_1D_atmos	19400	1	View Report
Allocator sizeof operand mismatch	src/cloud.c	read_ECMWF_clouds	5058	1	View Report
Allocator sizeof operand mismatch	libsrc_c/mystic.c	setup_caoth3D	7523	1	View Report
Argument with 'nonnull' attribute passed null	src/mie_lex.c	yytext2double	2887	18	View Report
Argument with 'nonnull' attribute passed null	src/mie_lex.c	yytext2float	2687	18	View Report
Argument with 'nonnull' attribute passed null	src/mie_lex.c	yytext2int	2790	18	View Report
Argument with 'nonnull' attribute passed null	libsrc_c/ascii.c	ASCII_readfile_nrows	1900	26	View Report
Argument with 'nonnull' attribute passed null	src/cldgen_lex.c	yytext2string	2251	23	View Report
Argument with 'nonnull' attribute passed null	src/cloud.c	read_and_convert_caoth_file	2422	24	View Report
Argument with 'nonnull' attribute passed null	src/albedo.c	read_AMBRALS_spectral_BRDFs	1351	58	View Report
Argument with 'nonnull' attribute passed null	src/mie_lex.c	yytext2string	2812	18	View Report
Argument with 'nonnull' attribute passed null	libsrc_c/ascii.c	ASCII_readfile_nrows	1905	33	View Report
Argument with 'nonnull' attribute passed null	src/cloud.c	read_and_convert_caoth_file	2514	24	View Report
Argument with 'nonnull' attribute passed null	src/cldgen_lex.c	yytextCstring	2273	23	View Report
Argument with 'nonnull' attribute passed null	src/albedo.c	read_RPV_spectral_BRDFs	1239	50	View Report
Argument with 'nonnull' attribute passed null	src/cldgen_lex.c	yytext2int	2233	23	View Report

- doxygen — generates automated documentation
- various compilers with warnings (gcc, icc, nag, aocc)
- learn to use debuggers (e.g. gdb, ipdb)
- static code analyzers (e.g. clang scan-build)

How to work with large codebases?

```
879         yy_current_state = yy_nxt[yy_base[yy_current_state] + yy_c];
880         ++yy_cp;
881       } while (yy_base[yy_current_state] != 975);
883     yy_find_action:
884       yy_act = yy_accept[yy_current_state];
885       if (yy_act == 0) { /* have to back up */
```

8 ← Assuming 'yy_act' is not equal to 0 →

9 ← Taking false branch →

```
891       YY_DO_BEFORE_ACTION;
893     do_action: /* This label is used only to access EOF actions. */
895       switch (yy_act) { /* beginning of action switch */
```

10 ← Control jumps to 'case 23:' at line 1238 →

```
1242            Input.wl.wl_step = yytext2double(yytext, 2, 2);
```

11 ← Calling 'yytext2double' →

```
2866 double yytext2double(char* yytext, int which_token, int max_token) {
2867   double dnumber = 0;
2868   char   token[] = " ";
2869   char * p = NULL, *s = NULL;
```

12 ← 's' initialized to a null pointer value →

```
2870   int    count     = 0;
2871   char*  tmpstring = NULL;
2873   tmpstring = (char*)calloc(strlen(yytext) + 1, sizeof(char));
2874   strcpy(tmpstring, yytext);
2876   p     = strtok(tmpstring, token);
2877   count = 1;
2878   while ((p = strtok(NULL, token)) != NULL) {
```

13 ← Assuming the condition is false →

14 ← Loop condition is false. Execution continues on line 2884 →

```
2884   if ((count != which_token) && (count > max_token))
```

15 ← Taking false branch →

```
2887   dnumber = atof(s);
```

16 ← Null pointer passed as an argument to a 'nonnull' parameter

- doxygen — generates automated documentation
- various compilers with warnings (gcc, icc, nag, aocc)
- learn to use debuggers (e.g. gdb, ipdb)
- static code analyzers (e.g. clang scan-build)

How to work with large codebases?

```
All make_angresfunc tests succeeded.
angres test
angres (0.00001,   0.1).....................................================
==3711==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61b000002238 at pc 0x5644116ee4fd bp
READ of size 8 at 0x61b000002238 thread T0
    #0 0x5644116ee4fc in main /builds/ls-mayer/libRadtran/src/angres.c:684
    #1 0x7fda16231b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #2 0x5644116e8439 in _start (/builds/ls-mayer/libRadtran/bin/angres+0x26439)

0x61b000002238 is located 0 bytes to the right of 1464-byte region [0x61b000001c80,0x61b000002238)
allocated by thread T0 here:
    #0 0x7fda17e3fd38 in __interceptor_calloc (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xded38)
    #1 0x5644116fe217 in ASCII_column /builds/ls-mayer/libRadtran/libsrc_c/ascii.c:2426
    #2 0x5644116ee43d in main /builds/ls-mayer/libRadtran/src/angres.c:683
    #3 0x7fda16231b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)

SUMMARY: AddressSanitizer: heap-buffer-overflow /builds/ls-mayer/libRadtran/src/angres.c:684 in main
Shadow bytes around the buggy address:
  0x0c367fff83f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fff8400: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fff8410: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fff8420: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c367fff8430: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c367fff8440: 00 00 00 00 00 00 00 00[fa]fa fa fa fa fa fa fa
  0x0c367fff8450: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8460: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8470: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8480: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c367fff8490: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
==3711==ABORTING
angres failed at test.pl line 1550.
        main::angres_test(0, 0.001, 1e-05) called at test.pl line 120
```

- doxygen — generates automated documentation
- various compilers with warnings (gcc, icc, nag, aocc)
- learn to use debuggers (e.g. gdb, ipdb)
- static code analyzers (e.g. clang scan-build)
- adress sanitizers and Valgrind

How to work with large codebases?

Filename	Line Coverage		Functions	
GLmystic.c	0.0 %	0 / 1018	0.0 %	0 / 31
alis.c	58.2 %	89 / 153	83.3 %	5 / 6
allocnd.c	82.6 %	19 / 23	22.0 %	20 / 91
ambralsfor.c	0.0 %	0 / 70	0.0 %	0 / 7
ascii.c	50.3 %	508 / 1009	63.3 %	50 / 79
bandec.c	49.4 %	43 / 87	50.0 %	2 / 4
c_tzs.c	57.9 %	162 / 280	50.0 %	2 / 4
cdisort.c	79.8 %	3165 / 3965	85.9 %	73 / 85
cnv.c	56.1 %	64 / 114	50.0 %	1 / 2
complex_surface.c	0.0 %	0 / 80	0.0 %	0 / 7
equation.c	46.3 %	138 / 298	50.0 %	3 / 6
errors.c	50.0 %	6 / 12	50.0 %	2 / 4
fortran_and_c.c	33.0 %	38 / 115	38.5 %	5 / 13
function.c	17.3 %	17 / 98	20.0 %	2 / 10
integrat.c	4.0 %	6 / 149	20.0 %	1 / 5
lidar.c	20.4 %	661 / 3235	33.3 %	15 / 45
linear.c	64.3 %	45 / 70	75.0 %	3 / 4
locate.c	93.2 %	41 / 44	100.0 %	2 / 2
miecalc.c	50.7 %	413 / 815	58.8 %	10 / 17
mystic.c	54.3 %	5062 / 9322	81.1 %	163 / 201
mystic_3d.c	63.5 %	94 / 148	100.0 %	2 / 2
nca.c	90.4 %	1068 / 1181	100.0 %	10 / 10
netCDF_functions.c	0.0 %	0 / 688	0.0 %	0 / 25
ocean.c	13.1 %	22 / 168	5.9 %	1 / 17
phasetable.c	67.3 %	491 / 730	83.3 %	20 / 24
rayleigh.c	46.1 %	35 / 76	50.0 %	2 / 4
raytracing.c	0.0 %	0 / 868	0.0 %	0 / 13
rodents.c	92.4 %	317 / 343	88.9 %	8 / 9
sos.c	1.4 %	4 / 295	12.5 %	1 / 8
specrend_uvspec.c	0.0 %	0 / 104	0.0 %	0 / 10
spl.c	16.4 %	56 / 341	25.0 %	2 / 8
sslidar.c	86.5 %	83 / 96	100.0 %	1 / 1
sun.c	4.2 %	8 / 191	13.3 %	2 / 15
sunpos.c	0.0 %	0 / 45	0.0 %	0 / 1
twomaxrnd.c	76.4 %	310 / 406	81.8 %	9 / 11
twomaxrnd3C.c	82.5 %	542 / 657	87.5 %	7 / 8
twostrebe.c	72.8 %	126 / 173	75.0 %	3 / 4
uvspecrandom.c	81.2 %	13 / 16	66.7 %	2 / 3
vroom.c	60.9 %	514 / 844	94.7 %	18 / 19

- doxygen — generates automated documentation
- various compilers with warnings (gcc, icc, nag, aocc)
- learn to use debuggers (e.g. gdb, ipdb)
- static code analyzers (e.g. clang scan-build)
- adress sanitizers and Valgrind
- gcov

How to work with large codebases?

```
Datei  Bearbeiten  Ansicht  Suchen  Terminal  Hilfe
✓   int doddis=0;
✓   int isp=0;
✓   int ispo=0;
✓   int isAmbralsFile=0;

✓   if ( sample->spherical3D ) {
✘     *sza  =  90.0 - sza_spher;
✘     *phi0 = -90.0 - phi0_spher;
    }

    /* define dimensions of scatter profiles concerning diffe
✓   atmos->nscaDS = MCSC_MODE_NORMAL+1;
✓   if (sample->delta_scaling!=-1)
✘     atmos->nscaDS = MCSC_MODE_DELTA_SCALE+1;

    /* define dimensions of scatter profiles concerning diffe
✓   atmos->nscaRIS = MCRIS_MODE_NORMAL+1;
#if HAVE_LIDAR
✓   if (sample->LLE_RIS_MAS || sample->RIS_MS)
✘     atmos->nscaRIS = MCRIS_MODE_MAS+1;
#ifdef NEWRISQIDD
      atmos->nscaRIS++;
#endif
#endif

    /* define dimensions of scatter profiles concerning diffe
✓   atmos->nscaVIS = MCVIS_MODE_NORMAL+1;
#if HAVE_LIDAR
✓   if (sample->LLE_VIS_QIDD)
✘     atmos->nscaVIS = MCVIS_MODE_QIDD+1;
#endif

    /* Rayleigh depolarisation */
✓   atmos->rayleigh_depol = rayleigh_depol;

✓   status = setup_profiles1D (n_caoth,
                  dt_s, om_s, gl_s, g2_s, f_s, ds_s,
                              re_s,
                              zprof, nlyr,
                  sample,
                              atmos,
                              alis);

✓   if (status!=0)
✘     return err_out ("Error %d returned by setup_profiles1D(

✓   if (!quiet)
```

- doxygen — generates automated documentation
- various compilers with warnings (gcc, icc, nag, aocc)
- learn to use debuggers (e.g. gdb, ipdb)
- static code analyzers (e.g. clang scan-build)
- adress sanitizers and Valgrind
- gcov

« Best Practices for Scientific Computing »

Wilson G et al. (2014) doi.org/10.1371/journal.pbio.1001745

- Let the computer do the work.
  - Make the computer repeat tasks.
  - Save recent commands in a file for re-use.
  - Use a build tool to automate workflows. **(remember climate model setup)**
- Make incremental changes.
  - Work in small steps with frequent feedback and course correction.
  - Use a version control system.
  - Put everything that has been created manually in version control. **(W2W data managment)**

« Best Practices for Scientific Computing »

Wilson G et al. (2014) doi.org/10.1371/journal.pbio.1001745

- Plan for mistakes.
  - Add assertions to programs to check their operation.
  - Use an off-the-shelf unit testing library. **(e.g. nosetest)**
  - Turn bugs into test cases.
  - Use a symbolic debugger.
- Collaborate.
  - Use pre-merge code reviews.
  - Use pair programming when bringing someone new up to speed and when tackling particularly tricky problems.
  - Use an issue tracking tool. **(gitlab merge requests and CI)**