HAMMOZ
AEROSOL & ATMOSPHERIC CHEMISTRY MODULES FOR ECHAM

JÜLICH
FORSCHUNGSZENTRUM

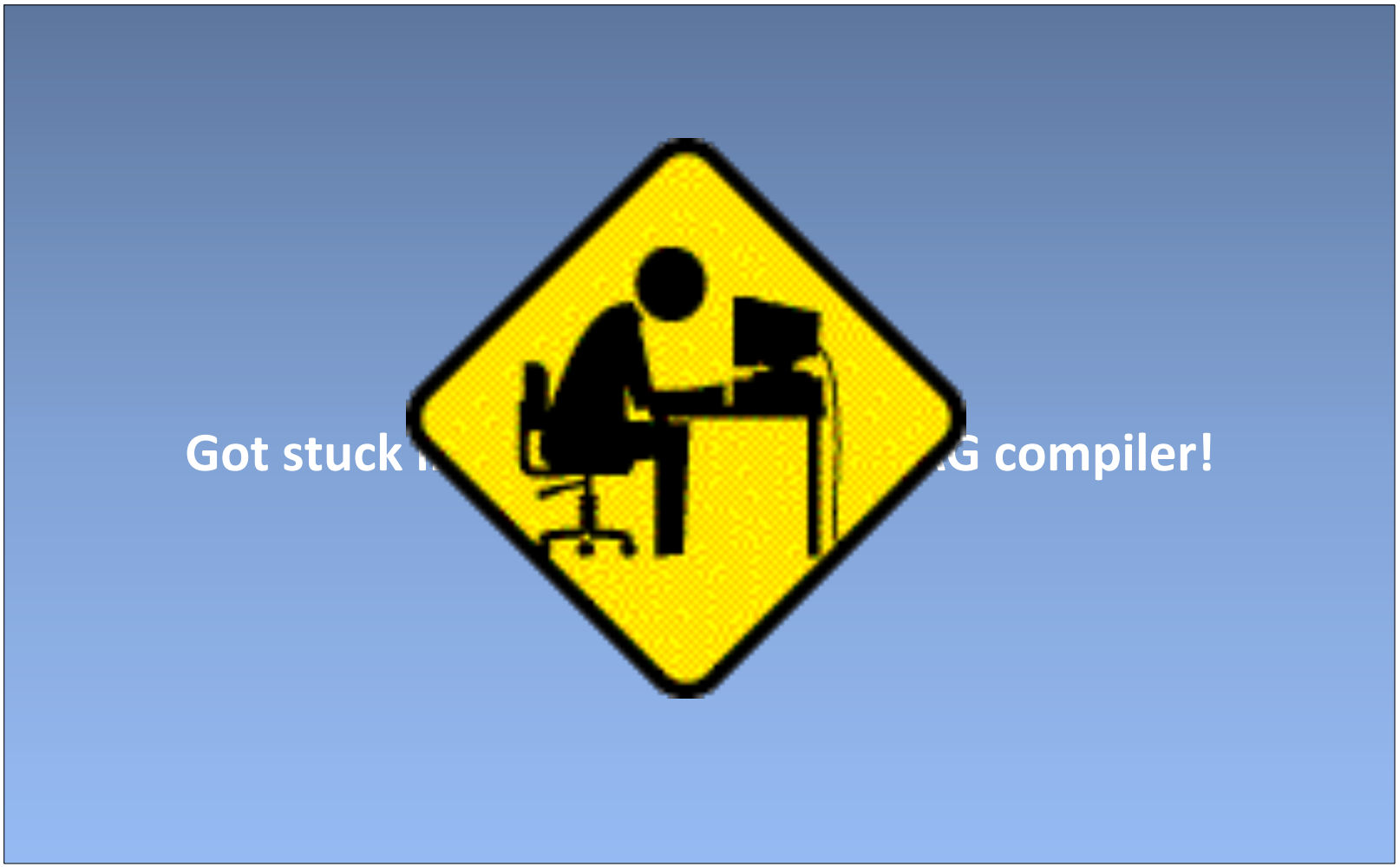
Notes on compiling and error checking

Sabine Schröder

Institute for Energy and Climate Research 8: Troposphere (IEK-8), Forschungszentrum Jülich, Germany

8th HAMMOZ workshop, Oxford, UK, 27-28 March 2014

Motivation



Motivation

Got stuck in compilation with NAG compiler!

Hammoz:
Use NAG!

Jülich IT service:
No support for NAG!

Motivation

To keep on working we had to find a „work-around“:

- Using other available compilers
- Finding out suitable compiling flags

configure settings

Options differ between vendors

General options:

- Which is intersection of all? What is compulsory?
- Which (if any) is ad hoc or individual?

Debugging/Testing options (NAG, GNU):

- What should be at minimum tested?
- What is just nice to have?

configure settings

Debugging FLAGS for NAG nagfor

-C=all: perform all checks except for *-C=undefined*

array (check array bounds)
bits (check bit intrinsic arguments)
calls (check procedure references)
dangling (check for dangling pointers)
do (check DO loops for zero step values)
present (check OPTIONAL references)
pointer (check POINTER references)
recursion (check for invalid recursion)

-float_store: Do not store floating-point variables in registers on machines with floating-point registers wider than 64 bits.

-nan: Initialise REAL and COMPLEX variables to IEEE Signalling NaN, causing a runtime crash if the values are used before being set.

-gline: Compile code to produce a traceback when a runtime error message is generated

-g: Produce information for interactive debugging

Default settings:
No optimisation
If no floating-point option is specified, any floating divide-by-zero, overflow or invalid operand exception will cause the execution of the program to be terminated

Compiling with available compilers

NAG nagfor	GNU gfortran	Intel ifort
-g	-g	-O0 -g
-C=array	-fbounds-check (-fcheck=bounds)	-CB
-C=bits → ??	→ ??	-assume noold_boz → ??
-C=calls	(per file) default	-gen-interfaces -warn interfaces
-C=dangling	--	(-check pointers)
-C=do	default (-fcheck=do)	--
-C=present	--	-warn interfaces
-C=pointer	-fcheck=pointer	-check pointers
-C=recursion	default (-fcheck=recursion)	default (when not nested)
-float_store	-ffloat-store	-fr32
floating point exceptions	-ffpe-trap=zero,overflow,invalid	-fpe0
-nan	-finit-real=nan -Wuninitialized	-ftrapuv (-CU)
-gline	default (-fbacktrace)	-traceback

Compiling with available compilers

JÜLICH FORSCHUNGSZENTRUM

Additional suggestions for NAG nagfor

-mtrace=all: Trace memory allocation and deallocation

- address (display addresses)
- line (display file/line info if known)
- on (enable tracing output)
- paranoia (protect memory allocator data structures against the user program)
- size (display size in bytes)

Additional suggestions for Intel ifort

-sox: Tells the compiler to save the compilation options and version number in the executable

-warn all
(alignments, usage, declarations, uncalled, unused)

Error checking with available compilers

Error detection by compiler/compiling flags

Example (array bounds error)

I) NAG

Runtime Error: mo_util_string.f90, line 33: Out of range: substring ending position 8 is greater than length 7
Program terminated by fatal error
mo_util_string.f90, line 33: Error occurred in MO_UTIL_STRING:TOLOWER
mo_namelist.f90, line 159: Called by MO_NAMELIST:POSITION_NML
test_bounds.f90, line 17: Called by TEST_BOUNDS

II) GNU

At line 33 of file mo_util_string.f90
Fortran runtime error: Substring out of bounds: upper bound (8) of 'upper' exceeds string length (7)

III) Intel

forrtl: severe (408): fort: (4): Variable UPPER has substring ending point 8 which is greater than the variable length of 7

Image	PC	Routine	Line	Source
test_bounds	000000000051D9ED	Unknown	Unknown	Unknown
…				
test_bounds	0000000000450D95	mo_util_string_mp	33	mo_util_string.f90
test_bounds	000000000045CA39	mo_namelist_mp_po	159	mo_namelist.f90
test_bounds	0000000000466D93	MAIN__	17	test_bounds.f90
…				

Error checking with available compilers

Importance of traceback:
Error detection by operating system
Example (hanging program due to missing broadcast)

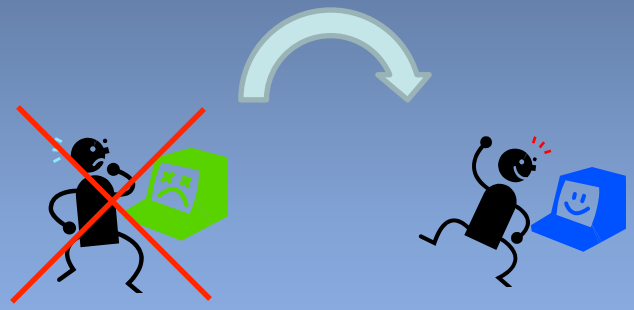
- varies from telling you nothing to verbose details
- traceback is helpful, if you get any information

Suggestions for testing/debugging in development phase:

- allow additional compilers for testing in development phase (e. g. PGI, GNU, Intel, Lahey)
- agree upon systematic set of compiling flags
- Development changes (source code) are accepted if successfully tested with two known compilers

Suggestions for compilation in production mode:

- review and harmonize compiling flags in configure settings



Thank you for your attention!