Thread Safety

International standards are now making it practicable for developers to write portable multi-threaded applications. Consequently there is an increasing demand for Library developers to produce software that is thread safe.

In a Fortran 77 context the constructs that prohibit thread safety are, potentially, DATA, SAVE, COMMON and EQUIVALENCE. This is because such constructs define data that will be shared by different threads, perhaps leading to unwanted interactions between them; for example, the possibility that one thread may be modifying the contents of a COMMON block at the same time as another thread is reading it. You are therefore advised to avoid the use of such constructs wherever possible within multi-threaded applications.

At Mark 19 of the NAG Library the use of unsafe constructs has been eliminated from the majority of routines in the Library, making them thread safe. However, there are some routines where complete removal of these constructs would seriously affect their interface design and usability. In such cases it makes more sense to keep the routines unchanged and give clear warnings in the documentation that care should be taken when calling such routines in a multi-threaded context. It should be noted that it is safe to call any NAG routine in one thread (only) of a multi-threaded application.

Some Library routines require you to supply a routine and to pass the name of the routine as an argument in the call to the Library routine. It is often the case that you need to supply your routine with more information than can be given via the interface argument list. In such circumstances it is usual to define a COMMON block containing the required data in the supplied routine (and also in the calling program). It is safe to do this only if no data referenced in the defined COMMON block is updated within the supplied routine (thus avoiding the possibility of simultaneous modification by different threads). Where separate calls are made to a Library routine by different threads and these calls require different data sets to be passed through COMMON blocks to user-supplied routines, these routines and the COMMON blocks defined within them should have different names.

You are also advised to check whether the Library routines you intend to call have equivalent reverse communication interfaces, which are designed specifically for problems where user-supplied routine interfaces are not flexible enough for a given problem; their use should eliminate the need to provide data through COMMON blocks.

The Library contains routines for setting the current error and advisory message unit numbers (X04AAF and X04ABF). These routines use the SAVE statement to retain the values of the current unit numbers between calls. It is therefore not advisable for different threads of a multi-threaded program to set the message unit numbers to different values. A consequence of this is that error or advisory messages output simultaneously may become garbled, and in any event there is no indication of which thread produces which message. You are therefore advised always to select the 'soft failure' mechanism without any error message (IFAIL = +1, see Section 2.3 of Essential Introduction) on entry to each NAG routine called from a multi-threaded application; it is then essential that the value of IFAIL is tested on return to the application.

A related problem is that of multiple threads writing to or reading from files. You are advised to make different threads use different unit numbers for opening files and to give these files different names (perhaps by appending an index number to the file basename). The only alternative to this is for you to protect each write to a file or unit number; for example, by putting each WRITE statement in a critical region.

You are also advised to refer to the Users' Note for details of whether the Library has been compiled in a manner that facilitates the use of multiple threads. Please note however that at Mark 19 the routines listed in the following table are not thread safe in any implementations.

C02AFF	C02AGF	C02AHF	C02AJF	C05NDF	C05PDF
D01AHF	D01EAF	D01FDF	D01GBF	D01GCF	D01GDF
D01JAF	D02BJF	D02CJF	D02EJF	D02GAF	D02GBF
D02HAF	D02HBF	D02JAF	D02JBF	D02KAF	D02KDF
D02KEF	D02LAF	D02LXF	D02LYF	D02LZF	D02MVF
D02MZF	D02NBF	D02NCF	D02NDF	D02NGF	D02NHF
D02NJF	D02NMF	D02NNF	D02NRF	D02NSF	D02NTF

D02NUF	D02NVF	D02NWF	D02NXF	D02NYF	D02NZF
D02PCF	D02PDF	D02PVF	D02PWF	D02PXF	D02PYF
D02PZF	D02QFF	D02QGF	D02QXF	D02QYF	D02QZF
D02RAF	D02SAF	D02XJF	D02XKF	D02ZAF	D03PCF
D03PDF	D03PEF	D03PFF	D03PHF	D03PJF	D03PKF
D03PLF	D03PPF	D03PRF	D03PSF	D03PUF	D03PVF
D03PWF	D03PXF	D03PZF	D03RAF	D03RBF	D05BDF
D05BEF	E02GBF	E04DGF	E04DJF	E04DKF	E04MFF
E04MGF	E04MHF	E04MZF	E04NCF	E04NDF	E04NEF
E04NFF	E04NGF	E04NHF	E04NKF	E04NLF	E04NMF
E04UCF	E04UDF	E04UEF	E04UFF	E04UGF	E04UHF
E04UJF	E04UNF	E04UQF	E04URF	E04XAF	F02FCF
F02FJF	F02HCF	F04YCF	F04ZCF	F08JKF	F08JXF
F11BAF	F11BBF	F11BCF	F11DCF	F11DEF	F11GAF
F11GBF	F11GCF	F11JCF	F11JEF	G01DCF	G01DHF
G01EMF	G01HBF	G01JDF	G03FAF	G03FCF	G05CAF
G05CBF	G05CCF	G05CFF	G05CGF	G05DAF	G05DBF
G05DCF	G05DDF	G05DEF	G05DFF	G05DHF	G05DJF
G05DKF	G05DPF	G05DRF	G05DYF	G05DZF	G05EGF
G05EHF	G05EJF	G05EWF	G05EYF	G05EZF	G05FAF
G05FBF	G05FDF	G05FEF	G05FFF	G05FSF	G05GAF
G05GBF	G05HDF	G07AAF	G07BEF	G07EAF	G07EBF
G08EAF	G08EBF	G08ECF	G08EDF	G10BAF	G13DCF
H02BBF	H02BFF	H02BVF	H02CBF	H02CCF	H02CDF
H02CEF	H02CFF	H02CGF	X04AAF	X04ABF	