NAG Fortran Library

Thread Safety

1 Multithreaded Applications and Thread Safety

A thread is a basic entity to which an operating system allocates CPU time. A thread has its own registers, stack and process resources. Threads provide a convenient way of allowing an application to maximise its usage of CPU resources in a system, especially in a multiple processor configuration. A routine is termed 'thread safe' if it can be called safely from two or more concurrently running threads.

The remainder of this document describes thread safety within the context of the NAG Fortran Library and provides guidelines for calling Library routines from multithreaded applications.

2 Thread Safety and the NAG Fortran Library

It is essential that you refer to the Users' Note for details of whether the Library has been compiled in a manner that facilitates the use of multiple threads. Also, your local site may have decided only to install a Library of thread safe routines; please contact your site installer for details of the installation.

2.1 Thread Safe Constructs

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 may 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 use such constructs with great care and to avoid their use wherever possible within multithreaded applications.

The NAG Fortran Library has addressed these issues by

eliminating unsafe constructs wherever possible;

providing equivalent thread safe routines with the same functionality where complete removal of unsafe constructs would affect interface design. Two approaches have been taken to provide thread safe equivalents; see Section 2.2 for further details.

2.2 Library Routines with Thread Safe Equivalents

In the NAG Fortran Library there are sometimes pairs of routines which share the same root name, for example, the routines D03PCF and D03PCA. Each routine in the pair has exactly the same functionality, except that one of them has additional parameters in order to make it safe for use in multithreaded applications. The routine that is safe for use in multithreaded applications has a different last character in the name in place of the usual character (typically 'A' instead of 'F'). Such pairs are documented via one routine document. If the pair of routines contain a routine argument in their interface then the routine with additional parameters will have parameter arrays that enable you to pass information to the routine argument without the need for COMMON blocks. In some cases the routine with additional parameters may need to be initialised by a separate initialisation routine; this requirement will be clearly documented.

2.3 Routines with Routine Arguments

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. For many of these Library routines, the supplied routine interface includes array arguments specifically for you to pass information to the supplied routine. However, there remain some Library routines for which you may need to supply your provided 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 advised to check, in the relevant chapter introduction, whether the Library routines you intend to call have equivalent reverse communication interfaces. These have been designed specifically for problems where user-supplied routine interfaces are not flexible enough for a given problem, and their use should eliminate the need to provide data through COMMON blocks.

2.4 Input/Output

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 multithreaded 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 the Essential Introduction) on entry to each NAG routine called from a multithreaded application; it is then essential that the value of IFAIL be 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.

2.5 Implementation Issues

In some implementations of the NAG Library calls are made to vendor BLAS and/or LAPACK Library routines. Where appropriate, NAG perform tests to ensure that these calls are behaving correctly on multiple threads, but cannot guarantee the thread safety of the vendor BLAS and LAPACK routines. You are advised to refer to the Users' Note for details of whether the Library is to be linked with vendor BLAS and/or LAPACK Libraries.