NAG Library Routine Document

X06AAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

X06AAF sets the number of threads requested for subsequent OpenMP parallel regions.

2 Specification

```
SUBROUTINE X06AAF (NUM, IFAIL)
INTEGER NUM, IFAIL
```

3 Description

X06AAF, for multi-threaded implementations, sets the number of threads to be requested for subsequent parallel regions to NUM. The first element of the list held by the OpenMP Internal Control Variable (ICV) used in determining the number of threads is set. See the Users' Note for your implementation for details of the scope of this routine.

The number of threads used in parallel regions will be equal to, or less than, the value of the ICV. The actual number of threads used is dependent on several factors, such as the presence of a `num_threads` clause on the `parallel` directive or the number of threads already in use by the program. Please refer to Section 4 for a full description of how the number of threads is chosen for a particular parallel region.

In serial implementations of the NAG Library this routine has no effect. See the X06 Chapter Introduction for a discussion of the behaviour of these routines when called in serial.

4 References

OpenMP Specifications http://openmp.org/wp/OpenMP-Specifications

Chapman B, Jost G and van der Pas R (2008) *Using OpenMP Portable Shared Memory Parallel Programming* The MIT Press

5 Arguments

1: NUM – INTEGER *Input*

On entry: the number of threads requested for subsequent OpenMP parallel regions.

Constraint: NUM ≥ 1.

2: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, NUM $= \langle value \rangle$.
Constraint: NUM ≥ 1.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

X06AAF is not threaded in any implementation.

9 Further Comments

None.

10 Example

In this example we presume a multi-threaded implementation of the NAG Library. We set the OpenMP Internal Control Variable used in determining the number of threads to 5 with a call to X06AAF and retrieve it again with X06ACF.

We then, using X06ABF, display the number of threads in use both outside, and inside, the OpenMP parallel region.

We expect to see X06ABF returning 1 outside of the parallel region, as the current team of threads there will consist of a single thread, and 5 from within it.

If you use a serial implementation of the NAG Library, regardless of whether the code is compiled with OpenMP or not, calling X06AAF has no effect and X06ABF and X06ACF will always return 1. The appropriate results file will be included with the distribution material for your implementation.

10.1 Program Text

```
    Program x06aafe

!     X06AAF Example Program Text

!     Mark 26 Release. NAG Copyright 2015.

!     .. Use Statements ..
      Use nag_library, Only: x06aaf, x06abf, x06acf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nout = 6
!     .. Local Scalars ..
      Integer                          :: ifail, num, num_max, num_set
!     .. Executable Statements ..
      Write (nout,*) 'X06AAF Example Program Results'
      Write (nout,*)

      ifail = 0
      num_set = 5

!     Set the OpenMP Internal Control Variable (ICV) controlling the number
!     of threads
      Call x06aaf(num_set,ifail)

!     Retrieve the value of the ICV, and display it
      num_max = x06acf()
      Write (nout,99999) 'Value of ICV controlling the number of threads:',    &
        num_max
      Write (nout,*)

!     Display the number of threads in the current team
      num = x06abf()
      Write (nout,99999) 'Number of threads outside the parallel region: ',    &
        num
      Write (nout,*)

!     Spawn an OpenMP parallel region and have the master thread display
!     the number of threads in the current team

      !$Omp Parallel Private (num), Default (None)

        num = x06abf()

        !$Omp Master
          Write (nout,99999) 'Number of threads inside the parallel region: ', &
            num
          Write (nout,*)
        !$Omp End Master

      !$Omp End Parallel

99999 Format (1X,A,I5)

    End Program x06aafe
```

10.2 Program Data

None.

10.3 Program Results

```
X06AAF Example Program Results

 Value of ICV controlling the number of threads:    5

 Number of threads outside the parallel region:     1

 Number of threads inside the parallel region:      5
```