NAG Library Routine Document

X06AGF

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

X06AGF enables or disables nested OpenMP parallelism.

2   Specification

```
SUBROUTINE X06AGF (NESTING)
INTEGER NESTING
```

3   Description

X06AGF, for multi-threaded implementations, enables or disables the nesting of OpenMP parallel regions by setting an OpenMP Internal Control Variable (ICV) and any vendor library specific options where that is possible. See the Users' Note for your implementation for details of the scope of X06AGF.

Nesting is disabled by default in OpenMP.

In serial implementations of the NAG Library X06AGF has no effect. See the Chapter X06 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:    NESTING – INTEGER   *Input*

On entry: if NESTING = 0, nesting of OpenMP parallel regions is disabled, otherwise it is enabled.

6   Error Indicators and Warnings

None.

7   Accuracy

Not applicable.

8   Parallelism and Performance

X06AGF 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 call X06AGF to enable the nesting of OpenMP parallel regions by setting an ICV. Inside a first, or 'outer', parallel region we call X06AHF to display the status of nesting, and expect to see it return the value 1.

Each thread then spawns a second 'inner' parallel region. Within this region we have each thread increment the value `total`. As there are 5 threads in the outer region and 3 threads in each inner region we expect the final value of this variable to be 15.

If you use a serial implementation of the NAG library, regardless of whether the code is compiled with OpenMP or not, X06AGF will have no effect and X06AHF will always return 0. The total number of threads will be 1. The appropriate results file will be included with the distribution material for your implementation.

10.1 Program Text

```
      Program x06agfe

!     X06AGF Example Program Text

!     Mark 26 Release. NAG Copyright 2015.

!     .. Use Statements ..
      Use nag_library, Only: x06aaf, x06agf, x06ahf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nout = 6
!     .. Local Scalars ..
      Integer                          :: ifail, nesting, nesting_set,     &
                                          num_inner, num_outer, total
!     .. Executable Statements ..
      Write (nout,*) 'X06AGF Example Program Results'
      Write (nout,*)

!     Enable the nesting of OpenMP parallel regions
      nesting_set = 1
      Call x06agf(nesting_set)

      ifail = 0
      num_inner = 3
      num_outer = 5

!     Set the number of OpenMP threads for an outer parallel region
      Call x06aaf(num_outer,ifail)

!     Spawn an OpenMP parallel region and have the master thread check whether
!     nesting of parallel regions has been enabled.

      total = 0

!$Omp Parallel Shared (num_inner,total), Private (ifail,nesting),     &
!$Omp   Default (None)

        nesting = x06ahf()

        !$Omp Master
          Write (nout,99999) 'Nesting enabled:', nesting
          Write (nout,*)
        !$Omp End Master

        ifail = 0
!       Set the number of OpenMP threads for an inner parallel region
        Call x06aaf(num_inner,ifail)

!       Spawn a nested parallel region
        !$Omp Parallel Shared (total), Default (None)
```

```
!         Add up the total number of threads in all teams
          !$Omp Atomic
            total = total + 1

        !$Omp End Parallel

      !$Omp End Parallel

      Write (nout,99999) 'Total number of threads requested:',          &
        num_outer*num_inner
      Write (nout,99999) 'Total number of threads from sum: ', total
      Write (nout,*)

99999 Format (1X,A,I5)

    End Program x06agfe
```

10.2 Program Data

None.

10.3 Program Results

```
X06AGF Example Program Results

Nesting enabled:     1

Total number of threads requested:   15
Total number of threads from sum:    15
```