NAG Library Function Document

nag_omp_get_thread_num (x06adc)

1 Purpose

nag_omp_get_thread_num (x06adc) returns the OpenMP thread number of the calling thread.

2 Specification

```
#include <nag.h>
#include <nagx06.h>
Integer nag_omp_get_thread_num ()
```

3 Description

nag_omp_get_thread_num (x06adc), for multi-threaded implementations, returns the calling OpenMP thread's unique thread number within the current team. The master thread will always return 0. The remaining threads will return a value between 1 and the value returned by nag_omp_get_num_threads (x06abc) less 1.

If this function is called from a sequential part of a multi-threaded program then it will return the value 0.

In serial implementations of the NAG C Library this function will always return 0. See the x06 Chapter Introduction for a discussion of the behaviour of these functions 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

None.

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

In this example we presume a multi-threaded implementation of the NAG C Library. We call nag_omp_get_thread_num (x06adc) both outside and inside an OpenMP active parallel region. Outside we expect a single thread to display the value 0. Inside the region we use the value to have only the master thread display the result.

We also call nag_omp_in_parallel (x06afc) inside and outside of the region. Outside we expect it to return 0, as we are not in an active parallel region, and inside we expect to see the value 1, indicating that the parallel region is an active one.

If you use a serial implementation of the NAG library, regardless of whether the code is compiled with OpenMP or not, nag_omp_get_num_threads (x06abc) will always return 1 and nag_omp_get_thread_num (x06adc) and nag_omp_in_parallel (x06afc) will always return 0. The appropriate results file will be included with the distribution material for your implementation.

10.1 Program Text

```
/* nag_omp_get_thread_num (x06adc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 25, 2014.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagx06.h>

int
main(void)
{

  /*  Scalars */
  Integer exit_status = 0;
  Integer in_para, me, num_got;
#ifdef _OPENMP
  Integer num_req = 5;
#endif

  /* Output preamble */
  printf("nag_omp_get_thread_num (x06adc)");
  printf(" Example Program Results\n\n");

  /*
   * nag_omp_get_thread_num (x06adc).
   * Get the OpenMP thread number
   */
  me = nag_omp_get_thread_num();
  printf("\n%s %11" NAG_IFMT " \n\n", "Thread number:           ", me);

  /*
   * nag_omp_in_parallel (x06afc).
   * Check whether we are in an active parallel region
   */
  in_para = nag_omp_in_parallel();
  printf("\n%s %11" NAG_IFMT " \n\n", "In active parallel region:", in_para);
  fflush(stdout);

  /*
   * Spawn an OpenMP parallel region, have the master thread display the
   * number of threads and check whether we are in an active parallel region
   */

  #pragma omp parallel num_threads(num_req) private(in_para, me, num_got) \
          default(none)
  {
```

```
    me = nag_omp_get_thread_num();
    /*
     * nag_omp_get_num_threads (x06abc).
     * Get the number of OpenMP threads in the current team
     */
    num_got = nag_omp_get_num_threads();
    in_para = nag_omp_in_parallel();

    if (me == 0)
      {
 printf("\n%s %11" NAG_IFMT " \n\n", "Number of threads:        ",
              num_got);
 printf("\n%s %11" NAG_IFMT " \n\n", "In active parallel region:",
        in_para);
      }
  }
  fflush(stdout);

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_omp_get_thread_num (x06adc) Example Program Results


Thread number:                      0


In active parallel region:          0


Number of threads:                  1


In active parallel region:          0
```