NAG Library Function Document

nag_omp_set_num_threads (x06aac)

1 Purpose

nag_omp_set_num_threads (x06aac) sets the number of threads requested for subsequent OpenMP parallel regions.

2 Specification

```
#include <nag.h>
#include <nagx06.h>
```

```
void nag_omp_set_num_threads (Integer num, NagError *fail)
```

3 Description

nag_omp_set_num_threads (x06aac), 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 function.

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 C Library this function has no effect. 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

1: **num** – Integer *Input*

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

Constraint: **num** ≥ 1.

2: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_INT

On entry, **num** = ⟨*value*⟩.
Constraint: **num** ≥ 1.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

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 Library. We set the OpenMP Internal Control Variable used in determining the number of threads to 5 with a call to nag_omp_set_num_threads (x06aac) and retrieve it again with nag_omp_get_max_threads (x06acc).

We then, using nag_omp_get_num_threads (x06abc), display the number of threads in use both outside, and inside, the OpenMP parallel region.

We expect to see nag_omp_get_num_threads (x06abc) 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 nag_omp_set_num_threads (x06aac) has no effect and nag_omp_get_num_threads (x06abc) and nag_omp_get_max_threads (x06acc) will always return 1. The appropriate results file will be included with the distribution material for your implementation.

10.1 Program Text

```
/* nag_omp_set_num_threads (x06aac) 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 num, num_max, num_set;

  /* Nag Types */
  NagError fail;

  INIT_FAIL(fail);

  /* Output preamble */
  printf("nag_omp_set_num_threads (x06aac)");
  printf(" Example Program Results\n\n");

  num_set = 5;

  /*
   * nag_omp_set_num_threads (x06aac).
   * Set the OpenMP Internal Control Variable (ICV) controlling the number
   * of threads
   */
  nag_omp_set_num_threads(num_set, &fail);

  if (fail.code != NE_NOERROR) {
    printf("Error from nag_omp_set_num_threads (x06aac).\n%s\n", fail.message);
    fflush(stdout);
    exit_status = 1;
    goto END;
  }

  /*
   * nag_omp_get_max_threads (x06acc).
   * Retrieve the value of the ICV
   */
  num_max = nag_omp_get_max_threads();
  printf("\n%s %11" NAG_IFMT " \n\n",
  "Value of ICV controlling the number of threads:", num_max);

  /*
   * nag_omp_get_num_threads (x06acc).
   * Get the number of threads in the current team
   */
  num = nag_omp_get_num_threads();

  printf("\n%s %11" NAG_IFMT " \n\n",
  "Number of threads outside the parallel region: ", num);

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

  #pragma omp parallel private (num) default(none)
  {
    num = x06abc();
    #pragma omp master
    {
      printf("\n%s %11" NAG_IFMT " \n\n",
      "Number of threads inside the parallel region:  ", num);
    }
  }
  fflush(stdout);

END:
  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_omp_set_num_threads (x06aac) Example Program Results

Value of ICV controlling the number of threads:           1

Number of threads outside the parallel region:            1

Number of threads inside the parallel region:             1
```