NAG Library Chapter Introduction

x06 – OpenMP Utilities

Contents

1 Scope of the Chapter

This chapter contains utilities for controlling the OpenMP environment for your program. They are based on OpenMP runtime library routines, although their functionality varies slightly.

2 Background to the Problems

These functions have been designed to be used with multi-threaded implementations of the NAG C Library. In these implementations, these functions enable you to change and interrogate the OpenMP threading environment for your whole program. In describing their use we assume you have followed the recommendations in the Users' Note. Functions are provided to control the number of threads, test whether you have active threads, get a thread's unique thread number and enable and disable nested parallelism. Readers are directed to How to Use the NAG Library and its Documentation for a wider discussion on parallelism.

As these functions apply to the whole program they will affect the OpenMP in your calling program, OpenMP used internally in the NAG C Library and also multi-threading in any underlying vendor libraries, where provided. See the Users' Note of your implementation for information on what underlying libraries have been used and for the scope of the X06 functions.

OpenMP uses the notion of Internal Control Variables (ICVs) to control the behaviour of a multi-threaded program. There are only two that are relevant to this chapter. One is used in determining the number of threads and the other controls the nesting of parallel regions. The user does not have direct access to ICVs, but they can be changed or reported with a call to an X06 function.

3 Recommendations on Choice and Use of Available Functions

The function nag_using_threaded_impl (x06xac) can be used to determine, at runtime, whether you are using a multi-threaded implementation of the NAG C Library or not.

3.1 Multi-threaded Implementations of the NAG Library

If you are not using OpenMP in your program we recommend you set the number of threads with the `OMP_NUM_THREADS` environment variable as described in the Users' Note. This is the number of threads that will then be used in multi-threaded NAG C Library functions. The ICV is set from this environment variable but the value can be changed with a call to nag_omp_set_num_threads (x06aac). It applies to the **next** parallel region, whether that is your own, one encountered by a NAG function or an underlying vendor library routine.

Whilst the ICV strongly influences the number of threads used, the design of OpenMP is such that it does not dictate it. Many factors affect the number of threads in a particular parallel region including, but not restricted to, the presence of a `num_threads` clause and the number of threads already in use by a program. However, in most cases the number of threads requested will be used. The value of the ICV is retrieved with a call to nag_omp_get_max_threads (x06acc). The return value is an upper bound on the number of threads. If it is crucial to know the number of threads that are actually in use for a particular parallel region we recommend you get this number with a call to nag_omp_get_num_threads (x06abc), once you are inside the parallel region.

OpenMP threads have a unique thread number, which can be retrieved for a particular thread by a call to nag_omp_get_thread_num (x06adc). The master thread is always numbered 0.

To check whether you are in an active parallel region, where there is more than one thread, nag_omp_in_parallel (x06afc) can be used.

The functions nag_omp_get_num_threads (x06abc), nag_omp_get_thread_num (x06adc) and nag_om p_in_parallel (x06afc) are only relevant when called from within an OpenMP parallel region. This could be one of your own or one in a NAG function. The cases where these routines apply to NAG functions are the ones which take a user-supplied function. There are functions in Chapters d01, d03, e05 and f01 which contain parallel regions that have calls to user-supplied functions from within them. You may, for example, wish to know the thread number, the number of threads or simply check whether this NAG parallel region is an active one in your supplied function.

Nested parallelism is where a parallel region is contained within another. That is, each thread in the **outer** region spawns its own **inner** parallel region of which it is the master thread. nag_omp_set_nested (x06agc) can be used to enable nested parallelism by setting the nesting ICV. nag_omp_get_nested (x06ahc) can be used to retrieve the value of this ICV. Nesting will be disabled by default and you should have a good reason for using nested parallel regions with careful thought given to the hardware resources you have.

If you wish to call a NAG multi-threaded function and have it execute in parallel from each thread in your own parallel region you will need to enable nested parallelism. If you do not enable it the NAG function will simply execute in serial. When using nesting the environment variable `OMP_NUM_THREADS` can be given a comma-separated list of integers representing the number of threads you wish to use at each level of parallelism. Recall that nag_omp_set_num_threads (x06aac) can be used to set the number of threads for the **next** parallel region. To change the number of threads for a NAG function in this scenario, you would call nag_omp_set_num_threads (x06aac) once inside your own parallel region.

3.2 Serial Implementations of the NAG C Library

When using a serial implementation of the NAG C Library the X06 functions return a value in line with your whole program being executed in serial. This is irrespective of what `OMP_NUM_THREADS` has been set to or if you have compiled your program with OpenMP.

Table 1 shows the behaviour of these functions in serial implementations of the NAG C Library.

Note that underlying vendor libraries may still be using multi-threading. Check the Users' Note document of your implementation.

If you are using OpenMP in your code together with a serial implementation of the NAG C Library, we recommend you use the OpenMP runtime library routines directly to control threading in your program.

Function	Behaviour when called from a serial implementation of the NAG C Library
nag_omp_set_num_threads (x06aac)	No effect
nag_omp_get_num_threads (x06abc)	Returns 1
nag_omp_get_max_threads (x06acc)	Returns 1
nag_omp_get_thread_num (x06adc)	Returns 0
nag_omp_in_parallel (x06afc)	Returns 0
nag_omp_set_nested (x06agc)	No effect
nag_omp_get_nested (x06ahc)	Returns 0

4 Functionality Index

5 Auxiliary Functions Associated with Library Function Arguments

None.

6 Functions Withdrawn or Scheduled for Withdrawal

None.