OPENMP

В презентации использованы материалы книги А.С. Антонова «Параллельное программирование с использованием технологии OpenMP»

Демидовский университет

Директива

#pragma omp parallel [опция[[,] опция]...]

задает параллельную область.

Возможные опции:

If (условие)

num_threads (целочисленное выражение)

default(shared|none)

private(список)

firstprivate(список)

shared(список)

copyin(список)

reduction(оператор:список)

Демидовский университет

Функции

double omp_get_wtime ( );

возвращает в вызвавшей нити астрономическое время в секундах (вещественное число двойной точности), прошедшее с некоторого момента в прошлом.

double omp_get_wtick ( );

возвращает в вызвавшей нити разрешение таймера в секундах.

Переменные среды и вспомогательные функции

- Количество нитей, выполняющих параллельную область, задается переменной среды OMP_NUM_THREADS.

Функции

omp_set_num_threads (int num);

int omp_get_num_threads ();

позволяют задать и считать значение переменной OMP_NUM_THREADS.

Переменные среды и вспомогательные функции

- Возможность динамически изменять количество нитей, используемых для выполнения параллельной области, задается переменной среды OMP_DYNAMIC.

Функции

omp_set_dynamic (int num);

int omp_get_dynamic ( );

позволяют задать и считать значение переменной OMP_DYNAMIC.

Переменные среды и вспомогательные функции

Функция

int omp_get_max_threads ( );

возвращает максимально допустимое число нитей для использования в следующей параллельной области.

Функция

int omp_get_num_procs ( );

возвращает количество процессоров, доступных для использования программе пользователя на момент вызова.

Переменные среды и вспомогательные функции

- Переменная среды OMP_NESTED, управляет возможностью вложения параллельных областей.

Функции

omp_set_nested (int num);

int omp_get_nested ( );

позволяют задать и считать значение переменной OMP_NESTED.

Пример

```
#include <stdio.h>
#include <omp.h>
int main(int argc, char *argv[])
{
  int n;
  omp_set_nested(1);
  #pragma omp parallel private(n)
  {
        n=omp_get_thread_num();
        #pragma omp parallel
        {
              printf("Часть 1, нить %d - %d\n", n, omp_get_thread_num());
        }
  }
```

Демидовский университет

Пример

```
omp_set_nested(0);
#pragma omp parallel private(n)
{
        n=omp_get_thread_num();
        #pragma omp parallel
        {
                printf("Часть 2, нить %d - %d\n", n, omp_get_thread_num());
        }
 }
}
```

Демидовский университет

omp_in_parallel()

Функция

int omp_in_parallel(void);

возвращает 1, если она была вызвана из активной параллельной области программы.

Пример

```
#include <stdio.h>
#include <omp.h>
void mode(void)
{
  if(omp_in_parallel())
          printf("Параллельная область\n");
  else printf("Последовательная область\n");
 }
```

Демидовский университет

Пример

```
int main(int argc, char *argv[])
{
  mode();
  #pragma omp parallel
  {
        #pragma omp master
        {
              mode();
        }
   }
}
```

Демидовский университет

Директива single

Директивой single выделяется участок кода в параллельной области, который должен быть выполнен только один раз.

Возможные опции:

private(список)

firstprivate(список)

copyprivate(список)

nowait

Пример

```
#include <stdio.h>
#include <omp.h>
int main(int argc, char *argv[])
{
  #pragma omp parallel
  {
        printf("Сообщение 1\n");
        #pragma omp single nowait
        {
              printf("Одна нить\n");
        }
        printf("Сообщение 2\n");
  }
}
```

Демидовский университет

Пример

```
#include <stdio.h>
#include <omp.h>
int main(int argc, char *argv[])
{
  int n;
  #pragma omp parallel private(n)
  {
        n=omp_get_thread_num();
        printf("Значение n (начало): %d\n", n);
        #pragma omp single copyprivate(n)
        {     n=100;    }
        printf("Значение n (конец): %d\n", n);
  }
}
```

Директива master

Директива master определяет участок кода, который будет выполнен только нитью-мастером.

#pragma omp master

Пример

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  int n;
  #pragma omp parallel private(n)
  {
        n=1;
        #pragma omp master
        { n=2; }
        printf("Первое значение n: %d\n", n);
        #pragma omp barrier
        #pragma omp master
        { n=3; }
        printf("Второе значение n: %d\n", n);
  }
 }
```

Демидовский университет

Демидовский
университет