

**Интернет Университет
Суперкомпьютерных технологий**

**Система поддержки выполнения OpenMP-
программ. Переменные окружения,
управляющие выполнением OpenMP-
программы**

Учебный курс

**Параллельное программирование с
OpenMP**

Бахтин В.А., кандидат физ.-мат. наук,
заведующий сектором,
Институт прикладной математики им.
М.В.Келдыша РАН

Содержание

- ❑ Внутренние переменные, управляющие выполнением OpenMP-программы (ICV-Internal Control Variables).
- ❑ Задание/опрос значений ICV-переменных.
- ❑ Функции работы со временем.

Internal Control Variables.

Для параллельных областей:

- nthreads-var*
- thread-limit-var*
- dyn-var*
- nest-var*
- max-active-levels-var*

Для циклов:

- run-sched-var*
- def-sched-var*

Для всей программы:

- stacksize-var*
- wait-policy-var*

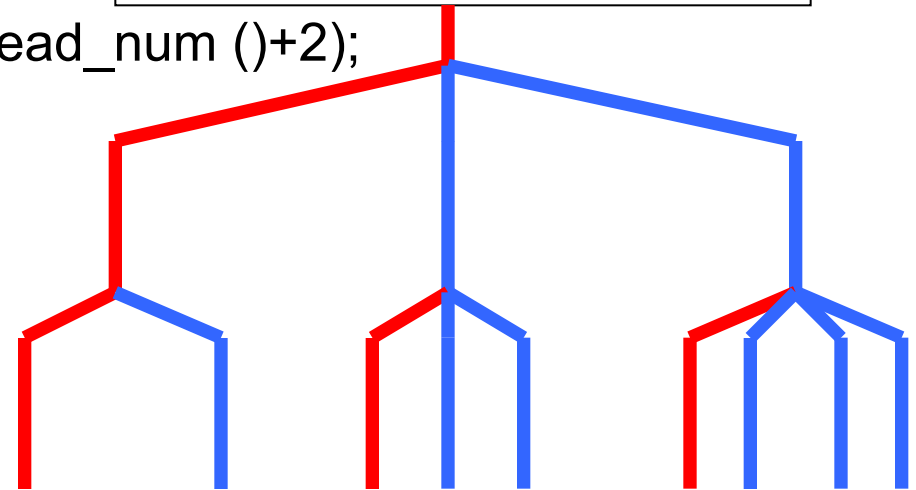
Internal Control Variables. *nthreads-var*

```
void work();
```

```
int main () {  
    omp_set_num_threads(3);  
    #pragma omp parallel  
    {  
        omp_set_num_threads(omp_get_thread_num()+2);  
        #pragma omp parallel  
        work();  
    }  
}
```

Не корректно в OpenMP 2.5

Корректно в OpenMP 3.0



Internal Control Variables. *nthreads-var*

Определяет максимально возможное количество нитей в создаваемой параллельной области.

Начальное значение: зависит от реализации.

Существует одна копия этой переменной для каждой задачи.

Значение переменной можно изменить:

❑ C shell:

```
setenv OMP_NUM_THREADS 16
```

❑ Korn shell:

```
export OMP_NUM_THREADS=16
```

❑ Windows:

```
set OMP_NUM_THREADS=16
```

```
void omp_set_num_threads(int num_threads);
```

Узнать значение переменной можно:

```
int omp_get_max_threads(void);
```

Internal Control Variables. *thread-limit-var*

Определяет максимальное количество нитей, которые могут быть использованы для выполнения всей программы.

Начальное значение: зависит от реализации.

Существует одна копия этой переменной для всей программы.

Значение переменной можно изменить:

❑ C shell:

```
setenv OMP_THREAD_LIMIT 16
```

❑ Korn shell:

```
export OMP_THREAD_LIMIT=16
```

❑ Windows:

```
set OMP_THREAD_LIMIT=16
```

Узнать значение переменной можно:

```
int omp_get_thread_limit(void)
```

Internal Control Variables. *dyn-var*

Включает/отключает режим, в котором количество создаваемых нитей при входе в параллельную область может меняться динамически.

Начальное значение: Если компилятор не поддерживает данный режим, то false. Иначе – зависит от реализации.

Существует одна копия этой переменной для каждой задачи.

Значение переменной можно изменить:

❑ C shell:

```
setenv OMP_DYNAMIC true
```

❑ Korn shell:

```
export OMP_DYNAMIC=true
```

❑ Windows:

```
set OMP_DYNAMIC=true
```

```
void omp_set_dynamic(int dynamic_threads);
```

Узнать значение переменной можно:

```
int omp_get_dynamic(void);
```

Internal Control Variables. *nest-var*

Включает/отключает режим поддержки вложенного параллелизма.

Начальное значение: **false**.

Существует одна копия этой переменной для каждой задачи.

Значение переменной можно изменить:

❑ C shell:

setenv OMP_NESTED true

❑ Korn shell:

export OMP_NESTED=false

❑ Windows:

set OMP_NESTED=true

void omp_set_nested(int nested);

Узнать значение переменной можно:

int omp_get_nested(void);

Internal Control Variables. *max-active-levels-var*

Задаёт максимально возможное количество активных вложенных параллельных областей.

Начальное значение: зависит от реализации.

Существует одна копия этой переменной для всей программы.

Значение переменной можно изменить:

❑ C shell:

```
setenv OMP_MAX_ACTIVE_LEVELS 2
```

❑ Korn shell:

```
export OMP_MAX_ACTIVE_LEVELS=3
```

❑ Windows:

```
set OMP_MAX_ACTIVE_LEVELS=4
```

```
void omp_set_max_active_levels (int max_levels);
```

Узнать значение переменной можно:

```
int omp_get_max_active_levels(void);
```

Internal Control Variables. *run-sched-var*

Задаёт способ распределения витков цикла между нитями, если указана клауза **schedule(runtime)**.

Начальное значение: зависит от реализации.

Существует одна копия этой переменной для каждой задачи.

Значение переменной можно изменить:

❑ C shell:

setenv OMP_SCHEDULE "guided,4"

❑ Korn shell:

export OMP_SCHEDULE "dynamic,5"

❑ Windows:

set OMP_SCHEDULE=static

```
typedef enum omp_sched_t {  
    omp_sched_static = 1,  
    omp_sched_dynamic = 2,  
    omp_sched_guided = 3,  
    omp_sched_auto = 4  
} omp_sched_t;
```

void omp_set_schedule(omp_sched_t kind, int modifier);

Узнать значение переменной можно:

void omp_get_schedule(omp_sched_t * kind, int * modifier);

Internal Control Variables. *run-sched-var*

```
void work(int i);
```

```
int main () {  
    omp_sched_t schedules [] = {omp_sched_static, omp_sched_dynamic,  
        omp_sched_guided, omp_sched_auto};  
    omp_set_num_threads (4);  
    #pragma omp parallel  
    {  
        omp_set_schedule (schedules[omp_get_thread_num()],0);  
        #pragma omp parallel for schedule(runtime)  
        for (int i=0;i<N;i++) work (i);  
    }  
}
```

Internal Control Variables. *def-sched-var*

Задаёт способ распределения витков цикла между нитями по умолчанию.

Начальное значение: зависит от реализации.

Существует одна копия этой переменной для всей программы.

```
void work(int i);
```

```
int main () {  
    #pragma omp parallel  
    {  
        #pragma omp for  
        for (int i=0;i<N;i++) work (i);  
    }  
}
```

Internal Control Variables. *stack-size-var*

Каждая нить представляет собой независимо выполняющийся поток управления со своим счетчиком команд, регистровым контекстом и стеком.

Переменная ***stack-size-var*** задает размер стека.

Начальное значение: зависит от реализации.

Существует одна копия этой переменной для всей программы.

Значение переменной можно изменить:

- setenv OMP_STACKSIZE 2000500B**
- setenv OMP_STACKSIZE "3000 k "**
- setenv OMP_STACKSIZE 10M**
- setenv OMP_STACKSIZE " 10 M "**
- setenv OMP_STACKSIZE "20 m "**
- setenv OMP_STACKSIZE " 1G"**
- setenv OMP_STACKSIZE 20000**

Internal Control Variables. *stack-size-var*

```
int main () {  
    int a[1024][1024];  
    #pragma omp parallel private (a)  
    {  
        for (int i=0;i<1024;i++)  
            for (int j=0;j<1024;j++)  
                a[i][j]=i+j;  
    }  
}
```

icl /Qopenmp test.cpp
⇒ **Program Exception – stack overflow**

Linux: ulimit -a
ulimit -s <stacksize in Kbytes>

Windows: /F<stacksize in bytes>
-Wl,--stack, <stacksize in bytes>

setenv KMP_STACKSIZE 10m
setenv GOMP_STACKSIZE 10000

setenv OMP_STACKSIZE 10M

Internal Control Variables. *wait-policy-var*

Подсказка OpenMP-компилятору о желаемом поведении нитей во время ожидания.
Начальное значение: зависит от реализации.

Существует одна копия этой переменной для всей программы.

Значение переменной можно изменить:

```
setenv OMP_WAIT_POLICY ACTIVE
setenv OMP_WAIT_POLICY active
setenv OMP_WAIT_POLICY PASSIVE
setenv OMP_WAIT_POLICY passive
```

```
IBM AIX
SPINLOOPTIME=100000
YIELDLOOPTIME=40000
```

Internal Control Variables. Приоритеты

клауза	вызов функции	переменная окружения	ICV
	<code>omp_set_dynamic()</code>	<code>OMP_DYNAMIC</code>	<i>dyn-var</i>
	<code>omp_set_nested()</code>	<code>OMP_NESTED</code>	<i>nest-var</i>
<code>num_threads</code>	<code>omp_set_num_threads()</code>	<code>OMP_NUM_THREADS</code>	<i>nthreads-var</i>
<code>schedule</code>	<code>omp_set_schedule()</code>	<code>OMP_SCHEDULE</code>	<i>run-sched-var</i>
<code>schedule</code>			<i>def-sched-var</i>
		<code>OMP_STACKSIZE</code>	<i>stacksize-var</i>
		<code>OMP_WAIT_POLICY</code>	<i>wait-policy-var</i>
		<code>OMP_THREAD_LIMIT</code>	<i>thread-limit-var</i>
	<code>omp_set_max_active_levels()</code>	<code>OMP_MAX_ACTIVE_LEVELS</code>	<i>max-active-levels-var</i>



Система поддержки выполнения OpenMP-программ.

```
int omp_get_num_threads(void);
```

 возвращает количество нитей в текущей параллельной области

```
#include <omp.h>
void work(int i);
void test()
{
    int np;
    np = omp_get_num_threads(); /* np == 1*/
    #pragma omp parallel private (np)
    {
        np = omp_get_num_threads();
        #pragma omp for schedule(static)
        for (int i=0; i < np; i++)
            work(i);
    }
}
```

Система поддержки выполнения OpenMP-программ.

```
int omp_get_thread_num(void);
```

 возвращает номер нити в группе [0: omp_get_num_threads()-1]

```
#include <omp.h>
void work(int i);
void test()
{
    int iam;
    iam = omp_get_thread_num(); /* iam == 0*/
    #pragma omp parallel private (iam)
    {
        iam = omp_get_thread_num();
        work(iam);
    }
}
```

Система поддержки выполнения OpenMP-программ.

```
int omp_get_num_procs(void);
```



возвращает количество процессоров, на которых программа выполняется

```
#include <omp.h>
```

```
void work(int i);
```

```
void test()
```

```
{
```

```
    int nproc;
```

```
    nproc = omp_get_num_procs();
```

```
    #pragma omp parallel num_threads(nproc)
```

```
    {
```

```
        int iam = omp_get_thread_num();
```

```
        work(iam);
```

```
    }
```

```
}
```

Система поддержки выполнения OpenMP-программ.

```
int omp_get_level(void)
```

- возвращает уровень вложенности для текущей параллельной области.

```
#include <omp.h>
```

```
void work(int i) {
```

```
    #pragma omp parallel
```

```
    {
```

```
        int ilevel = omp_get_level ();
```

```
    }
```

```
}
```

```
void test()
```

```
{
```

```
    int ilevel = omp_get_level (); /*ilevel==0*/
```

```
    #pragma omp parallel private (ilevel)
```

```
    {
```

```
        ilevel = omp_get_level ();
```

```
        int iam = omp_get_thread_num();
```

```
        work(iam);
```

```
    }
```

```
}
```

Система поддержки выполнения OpenMP-программ.

```
int omp_get_active_level(void)
```

- возвращает количество активных параллельных областей (выполняемых 2-мя или более нитями).

```
#include <omp.h>
```

```
void work(int iam, int size) {
```

```
    #pragma omp parallel
```

```
    {
```

```
        int ilevel = omp_get_active_level ();
```

```
    }
```

```
}
```

```
void test()
```

```
{
```

```
    int size = 0;
```

```
    int ilevel = omp_get_active_level (); /*ilevel==0*/
```

```
    scanf("%d",&size);
```

```
    #pragma omp parallel if (size>10)
```

```
    {
```

```
        int iam = omp_get_thread_num();
```

```
        work(iam, size);
```

```
    }
```

```
}
```

Система поддержки выполнения OpenMP-программ.

int omp_get_ancestor_thread_num (int level)

- для нити, вызвавшей данную функцию, возвращается номер нити-родителя, которая создала указанную параллельную область.

omp_get_ancestor_thread_num (0) = 0

```
if (level==omp_get_level()) {  
    omp_get_ancestor_thread_num (level) == omp_get_thread_num ();  
}
```

```
if ((level<0)||level>omp_get_level()) {  
    omp_get_ancestor_thread_num (level) == -1;  
}
```

Система поддержки выполнения OpenMP-программ.

```
int omp_get_team_size(int level);
```

- количество нитей в указанной параллельной области.

```
omp_get_team_size (0) = 1
```

```
If (level==omp_get_level()) {  
    omp_get_team_size (level) == omp_get_num_threads ();  
}
```

```
If ((level<0)|| (level>omp_get_level())) {  
    omp_get_team_size (level) == -1;  
}
```

Система поддержки выполнения OpenMP-программ. Функции работы со временем

double omp_get_wtime(void);



возвращает для нити астрономическое время в секундах, прошедшее с некоторого момента в прошлом. Если некоторый участок окружить вызовами данной функции, то разность возвращаемых значений покажет время работы данного участка. Гарантируется, что момент времени, используемый в качестве точки отсчета, не будет изменен за время выполнения программы.

double start;

double end;

start = omp_get_wtime();

/... work to be timed ...*/*

end = omp_get_wtime();

printf("Work took %f seconds\n", end - start);

double omp_get_wtick(void);

- возвращает разрешение таймера в секундах (количество секунд между последовательными импульсами таймера).

Спасибо за внимание!

Вопросы?

Следующая тема

- **Наиболее часто встречаемые ошибки в OpenMP-программах. Функциональная отладка OpenMP-программ.**

Контакты

□ **Бахтин В.А.**, кандидат физ.-мат. наук, заведующий сектором, Институт прикладной математики им. М.В. Келдыша РАН

bakhtin@keldysh.ru