Programe de test pentru API OpenMP
în limbajul C
17-01-2018

1. Aplicaţie demonstrativă “Hello, world”

```
================================================================
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
	int nthreads, tid;
	//Se creeaza o serie de fire de executie avand variabile proprii
	//(nthreads si tid)

	//Sectiune de cod executata in paralel pe toate procesoarele
	#pragma omp parallel private(nthreads, tid)
	{

		//Se obtine identificatorul firului de executie
		tid = omp_get_thread_num();

		//Mesaj transmis de fiecare fir de executie
		printf("Hello World din firul:%d\n", tid);

		//Cod executat doar de firul 'master'
		if (tid == 0)
		{
			nthreads = omp_get_num_threads();
			printf("Number of threads = %d\n", nthreads);
		}
		//Toate firele de executie isi termina rularea si elibereaza resursele
	}

}
================================================================
```

2. Aplicaţie demonstrativă ce calculează media alunecătoare a doi vectori A şi B de mărime N, având o fereastră cu dimensiunea JMAX.

2.1 Versiunea cu un singur fir de execuţie

```
================================================================
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

#define	N		20000000

int main (int argc, char *argv[])
{
	//declaratii variabile
	int i,j,jmax;
	float a[N], b[N], c[N];

	//initializari
	for (i=0;i<N;i++)
	a[i]=b[i]=i*1.0;
	jmax=15;
	for (i=jmax;i<N-jmax;i++)
	{
		c[i]=0.0;
		for(j=0;j<2*jmax;j++)
		{
		c[i]=c[i]+(a[i-jmax+j]+b[i-jmax+j])/2*jmax;
		}
	}
}
================================================================
```

2.2 Versiunea cu mai multe fire de execuție

CHUNKSIZE = dimensiunea unei secțiuni elementare pentru care se execută un fir specific

```
================================================================
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#define        CHUNKSIZE      100
#define        N              20000000

int main (int argc, char *argv[])
{
        //declaratii variabile
        int nthreads, tid, i, j, jmax, chunk;
        float a[N], b[N], c[N];

        //initializari vectori si dimensiune sectiune 'chunk'
        for (i=0; i < N; i++)
        a[i] = b[i] = i * 1.0;
        chunk = CHUNKSIZE;
        jmax=15;
        //start sectiune de calcul paralel - variabile locale: i,j,tid
        #pragma omp parallel shared(a,b,c,nthreads,chunk) private(i,j,tid)
        {
                tid = omp_get_thread_num();
                if (tid == 0)
                {
                        nthreads = omp_get_num_threads();
                        printf("Numar fire de executie=%d\n",nthreads);
                }
                printf("Firul de executie %d porneste...\n",tid);

                #pragma omp for schedule(dynamic,chunk)
                for(i=jmax;i<N-jmax;i++)
                {
                        c[i]=0.0;
                        for(j=0;j<2*jmax;j++)
                        {
                        c[i]=c[i]+(a[i-jmax+j]+b[i-jmax+j])/2*jmax;
                        }
                }
        }
        //sfarsitul sectiunii de calcul paralel
}
================================================================
```

Compilarea fișierelor sursă se face cu ajutorul compilatorului GCC utilizând comanda:

```
> gcc       -fopenmp    fisier_sursa.c
```

În cazul compilării cu succes a fișierului sursă, se generează fișierul executabil denumit implicit: *a.out*.

Pentru evaluarea timpului de rulare al programelor realizate se utilizează comanda *time*.

```
> time ./a.out

real  0m3.168s
user  0m10.444s
sys   0m0.318s
```

Exerciţii

Ex 1. Modificaţi în sensul creșterii și descreșterii valorii CHUNKSIZE si urmăriţi efectul asupra timpului de rulare al aplicației.

Ex 2. Comparați timpii de rulare obținuți cu planificatorul “for schedule” setat *dynamic* sau *static*.