ORDIALES CABALLERO, IÑAKY

OpenMP resumen

Directivas o *pragma*:
    #pragma omp **nombreDelConstructor <clausula o clausulas>**

Compilación en consola:
    gcc -fopenmp nombreArchivo.c -o nombreDeseadoEjecutable

Ejecución en consola:
    ./nombreDeseadoEjecutable

Bilioioteca openmp:
    <omp.h>

Cambiar hilos desde consola:
    export OMP_NUM_THREADS=n

** Notas:

- Todas las variables declaradas dentro de una región paralela serán privadas, y declaradas fueras serán compartidas.
- **Race Condition:** el primero que llega lo usa antes, "compiten" → inconsistencia.
- **Región crítica:** porción de código que contiene actualización de variables compartidas (da lugar a race condition).
- **Exlusión mutua:** un hilo excluye temporalmente a todos los demás de un recurso compartido.

Función omp.h	USO
omp_set_num_threads(n)	Cambiar el número de hilos a n (un entero)
omp_get_num_threads()	Regresa el número de hilos activos.
omp_get_thread_num()	Regresa el número del hilo actual. (0 - n)

USO	Constructor	Cláusula(s)	Sintáxis

Crear regiones paralelas. (Fork – join)	**parallel**		#pragma omp **parallel**
			{
			//Bloque de código
			}

Cambiar número de hilos	parallel	**num_threads(n)**	#pragma omp parallel **num_threads(n)**
Las variables dentro serán compartidas.	parallel	**shared(v1, v2)**	#pragma omp parallel **shared(i)**
Variables serán priv. (crea copia por hilo y destruye al finalizar hilo.	parallel	**private(v1, v2)**	#pragma omp parallel **private(i)**

Debe estar dentro de una región paralela. Divide las iteraciones de una estructura for. La variable de control i, se hará privada.	**for**		#pragma omp parallel
			{
			…
			#pragma omp **for**
			for(i=0; i<n; i++) {
			Algo();
			}
			}

Se pueden agrupar / anidar los constructores parallel y for. Siempre y cuando solo se paralelice el ciclo for.	**parallel for**		#pragma omp **parallel for**

Al segmento crítico solo puede entrar un hilo a la vez.	**critical**		# pragma omp **critical**
			{
			//Código crítico
			}

Establece el número de hilos para el segmento paralelo.	parallel	**nthreads(n)**	#pragma omp parallel **nthreads(n)**
			{
			//Código en n hilos.
			}

Toma una variable dada de cada hilo y realiza una operación sobre estos datos. Regresando un solo resultado.	parallel	**reduction**
		
		(operaciones de reducción)
			#pragma omp parallel **reduction (operador:variable)**
			{
			
			}
			Ejemplo : reduction(+ : res)
			Suma todas las variables res de los hilos.

Se combinan cuando se sume la variable calculada de un ciclo.	**parallel for**	**reduction**	#pragma omp parallel for reduction(+ : a)

DESCOMPOSICIÓN FUNCIONAL
permite usar paralelismo funcional. Permite asignar secciones de código independientes a hilos diferentes para trabajar paralelo. (Tiene barrera implícita que sincroniza el final de las secciones)
	sections		#pragma omp parallel **sections**
			{
			#pragma omp **section**
			//Código sección 1
			#pragma omp **section**
			//Código sección 2
			#pragma omp **section**
			//Código sección 3
			}

Coloca una barrera explícita para que cada hilo espere a que todos lleguen a la barrera. Sincronizar. (Dependencias)	**barrier**		#pragma omp parallel
			{
			Algo();
			Otro();
			#pragma omp **barrier** // esperan
			Final();
			}

Define bloque de código dentro de sección paralela que debe ser ejecutado por solo un hilo, mientras los demás esperan.	**single**		#pragma omp parallel
			{
			todosRealizanAlgo();
			#pragma omp **single**
			{
			ActividadSolo();
			} //Hilos esperan
			todosMasActividades();
			}

La actividad única la realiza el hilo maestro (principal o main). Los demás hilos no esperan.	**master**		#pragma omp parallel
			{
			#pragma omp master
			{
			…
			}
			}

Quita las barreras implícitas de los constructores (for, single, sections).		**nowait**	

Generar tareas de forma explícita. Esas tareas pueden ser asignadas a otros hilos de la región.	**task**
	
	(Recursivo)
			#pragma omp **task**
			{
			
			}

Permite esperar que todas las tareas hijas generadas desde la actual terminen.	**taskwait**
	
	(Recursivo)
			#pragma omp **taskwait**
			{
			
			}