Recetario OpenMP

La sintanxis basica a la hora de decirle a openMP que tiene que hacer con el codigo que seleccionamos, es la siguiente:

```
# pragma omp <directiva> [cláusula [, … ] … ] {
    //codigo paralelizado
}
```

Directivas

Las directivas son instrucciones que se utilizan para indicar cómo paralelizar secciones específicas de un programa. Las directivas se escriben con comentarios precididas por #pragma:

- **parallel**: Indica que la ejecución se hará por varios hilos.

```
#pragma omp parallel
{
    // Código a ejecutar en paralelo
};
```

- **for**: Distribuye las iteraciones del bucle entre los hilos creados por la directiva omp parallel.

```
#pragma omp parallel
{
    #pragma omp for
    for (int i = 0; i < n; ++i) {
        // Código del bucle a ejecutar en paralelo
    }
}
```

- **sections:** Divide el código en secciones de código que se ejecutarán de forma paralela en diferentes hilos.

```
#pragma omp parallel
{
    #pragma omp sections
    {
        #pragma omp section
        {
            // Código de la primera sección
        }
        #pragma omp section
        {
            // Código de la segunda sección
        }
    }
}
```

- **critical:** Define una seccion del codigo que solo podra ser ejecutada por un hilo en ese momento, esto se usa para evitar condiciones de carrera.

```
#pragma omp parallel
{
    #pragma omp critical
    {
        // Código crítico
    }
}
```

- **barrier:** Pone un punto de espera hasta que todos los hilos hayan terminado sus trabajos.

```
#pragma omp barrier
```

- **atomic:** Sección de código que solo puede ser actualizada simultaneamente por un hilo.

```
#pragma omp atomic
```

- **single:** Indica que la sección de código solo podrá ser ejecutada por una sola hebra del equipo.

```
#pragma omp single
```

- **task:** Permite que se ejecuten tareas de forma asíncrona donde el código de la directiva task, es ejecutado por un único hilo.

```
#pragma omp parallel
{
    #pragma omp single
    {
        #pragma omp task
        {
            // Código de la tarea 1
        }
        #pragma omp task
        {
            // Código de la tarea 2
        }
    }
}
```

- **master:** El código seleccionado solo se podrá ejecutar por el hilo padre.

```
#pragma omp master
```

- **ordered:** Especifica que las iteraciones se ejecutarán en orden del bucle de forma secuencial.

```
#pragma omp parallel for ordered
```

- **threadprivate:** Especifica las variables que se replican, teniendo cada hebra una copia.

```
#pragma omp threadprivate(x)
```

Cláusulas

- **private:** La cláusula private permite declarar variables privadas para cada hilo.

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

- **firstprivate:** La cláusula firstprivate permite declarar variables privadas para cada hilo, inicializándolas con el valor que tenían antes de la región paralela.

```
#pragma omp parallel firstprivate(i)
```

- **lastprivate:** La cláusula lastprivate permite declarar variables privadas para cada hilo, inicializándolas con el valor que tenían al finalizar la región paralela.

```
#pragma omp parallel lastprivate(i)
```

- **shared:** La cláusula shared permite declarar variables compartidas por todos los hilos.

```
#pragma omp parallel shared(i)
```

- **default:** La cláusula default permite establecer el comportamiento por defecto de las variables, si no se especifica nada en la cláusula private o shared.

```
#pragma omp parallel default(shared)
```

- **reduction:**La cláusula reduction permite declarar variables privadas para cada hilo, inicializándolas con el valor que tenían antes de la región paralela.

```
#pragma omp parallel reduction(+:sum)
```

- **schedule:**La cláusula schedule permite establecer el reparto de iteraciones del bucle entre los hilos.

```
#pragma omp parallel for schedule(static, 1)
```

- **nowait:**La cláusula nowait permite indicar que no se debe esperar a que terminen los hilos para continuar con la ejecución del programa.

```
#pragma omp parallel for nowait
```

- **copyin:**La cláusula copyin permite copiar el valor de una variable privada de un hilo a una variable compartida.

```
#pragma omp parallel copyin(i)
```

- **copyprivate:**La cláusula copyprivate permite copiar el valor de una variable compartida a una variable privada de un hilo.

```
#pragma omp parallel copyprivate(i)
```

Schedule

- **static:**La cláusula schedule(static, n) reparte las iteraciones del bucle entre los hilos de forma estática, es decir, cada hilo se encarga de un bloque de n iteraciones.

```
#pragma omp parallel for schedule(static, 1)
```

- **dynamic:**La cláusula schedule(dynamic, n) reparte las iteraciones del bucle entre los hilos de forma dinámica, es decir, cada hilo se encarga de un bloque de n iteraciones, y cuando termina vuelve a por otro bloque de n iteraciones.

```
#pragma omp parallel for schedule(dynamic, 1)
```

- **guided:**La cláusula schedule(guided, n) reparte las iteraciones del bucle entre los hilos de forma dinámica, pero el tamaño del bloque de iteraciones que se asigna a cada hilo va disminuyendo a medida que se van asignando bloques.

```
#pragma omp parallel for schedule(guided, 1)
```

- **auto:**La cláusula schedule(auto) deja que el compilador decida el reparto de iteraciones del bucle entre los hilos.

```
#pragma omp parallel for schedule(auto)
```

- **runtime:**La cláusula schedule(runtime) deja que el reparto de iteraciones del bucle entre los hilos se decida en tiempo de ejecución.

```
#pragma omp parallel for schedule(runtime)
```

Variables de entorno

OMP_NUM_THREADS: Establece el numero de hilos que se van a utilizar en las regiones paralelas.
OMP_SCHEDULE: Establece el reparto de las iteraciones del bucle entre los hilos.
OMP_DYNAMIC: Establece si se puede asignar iteraciones de forma estática p dinámica.
OMP_NESTED: Establece si se pueden declarar regiones paralelas anidadas.
OMP_MAX_ACTIVE_LEVELS: Nos permite establecer un número máximo de funciones paralelas anidadas.
OMP_THREAD_LIMIT: Establece el máximo de hilos que se pueden usar.
OMP_STACKSIZE: Establece el tamaño de la pila en cada hilo.
OMP_WAIT_POLICY: Establece la política de espera de los hilos.
OMP_PROC_BIND: Establece si los hilos deben estar o no ligados a los nucleos del procesador.
OMP_PLACES: Establece la afinidad entre nucleos e hilos del procesador.
OMP_DISPLAY_ENV: Establece si se muestran o no las variables de entorno.
OMP_CANCELLATION: Establece si se permiten o no las cancelaciones de regiones paralelas.
OMP_DEFAULT_DEVICE: Establece el dispositivo por defecto para las regiones target.
OMP_MAX_TASK_PRIORITY: Establece la maxima prioridad entre las tareas.
OMP_TARGET_OFFLOAD: Establece si se deben usar dispositivos de aceleración.
OMP_TOOL: Establece si deben usarse las herramientas de OpenMP
OMP_ALLOCATOR: Establece el tipo de asignador de memoria.