Ejercicio 7 Planificación de Bucles

Programa 1

En este programa podemos ver la sentencia **"Schedule"** que sirve para definir estategias de reparto de tareas, en esta ocasión utiliza la sentencia **"static"** para indicar que divide el ciclo for en partes de k, y son repartidas por cada hilo, en este caso lo hace con 4, entonces el programa asigna una de esas partes de 4 por cada hilo que el programa tiene disponible y cuando lo hace en el ultimo hilo vuelve a empezar hasta que finaliza con el ciclo for.

Codigo:

```
#include <omp.h>
#include <stdio.h>
#include <unistd.h>
#define N 40
int main()
{
    int tid;
    int A[N];
    int i;
    for (i = 0; i < N; i++)
        A[i] = -1;
#pragma omp parallel for schedule(static, 4) private(tid)
    for (i = 0; i < N; i++)
    {
        tid = omp_get_thread_num();
        A[i] = tid;
        usleep(1);
    }
    for (i = 0; i < N / 2; i++)
        printf(" %2d", i);
    printf("\n");
    for (i = 0; i < N / 2; i++)
        printf(" %2d", A[i]);
    printf("\n\n\n");
    for (i = N / 2; i < N; i++)
        printf(" %2d", i);
    printf("\n");
    for (i = N / 2; i < N; i++)
        printf(" %2d", A[i]);
    printf("\n\n");
}
```

Salida:

```
  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
  0  0  0  0  1  1  1  1  2  2  2  2  3  3  3  3  4  4  4  4
```

```
 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
 5  5  5  5  6  6  6  6  7  7  7  7  0  0  0  0  1  1  1  1
```

Programa 2

En este caso el programa divide la iteracion for en partes de k igual que el programa anterior, pero en este caso las asigna dinamicamente a los hilos que tenemos disponibles, por lo tanto **El orden de la asignacion dependerá de la ejecución del programa**.

Codigo:

```
#include <omp.h>
#include <stdio.h>
#include <unistd.h>
#define N 40
int main()
{
    int tid;
    int A[N];
    int i;
    for (i = 0; i < N; i++)
        A[i] = -1;
#pragma omp parallel for schedule(dynamic, 4) private(tid)
    for (i = 0; i < N; i++)
    {
        tid = omp_get_thread_num();
        A[i] = tid;
        usleep(1);
    }
    for (i = 0; i < N / 2; i++)
        printf(" %2d", i);
    printf("\n");
    for (i = 0; i < N / 2; i++)
        printf(" %2d", A[i]);
    printf("\n\n\n");
    for (i = N / 2; i < N; i++)
        printf(" %2d", i);
    printf("\n");
    for (i = N / 2; i < N; i++)
        printf(" %2d", A[i]);
    printf("\n\n");

}
```

Salida:

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
 0  0  0  0  4  4  4  4  3  3  3  3  7  7  7  7  6  6  6  6

20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
 1  1  1  1  2  2  2  2  7  7  7  7  2  2  2  2  0  0  0  0
```