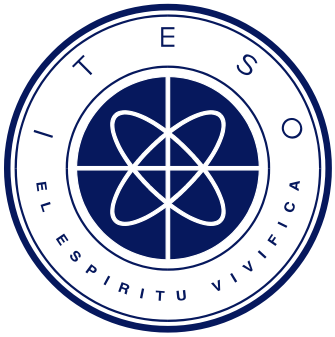
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA ELECTRÓNICA



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 2. APUNTADORES A FUNCIONES

Autor: Galeana Torres, Fernanda

Presentación: 10 pts.
Funcionalidad: 60 pts.
Pruebas: 20 pts.

11 de junio de 2018. Tlaquepaque, Jalisco,

- Las figuras deben tener un número y descripción.
- Las figuras, tablas, diagramas y algoritmos en un documento, son material de apoyo para transmitir ideas.
- Sin embargo deben estar descritas en el texto y hacer referencia a ellas. Por ejemplo: En la Figura 1….
- Falta describir las pruebas (escenario, y resultados de la experimentación).
- Cuando se tienen resultados que se pueden comparar, se recomienda hacer uso de diagramas o tablas que permitan observar el resultado de los diversos casos y contrastas los resultados (en el tiempo por ejemplo).

Objetivo de la actividad

El objetivo de la tarea es que el alumno aplique los conocimientos y habilidades adquiridos en el tema de apuntadores a funciones y la distribución de tareas mediante el suo de hilos para la resolución de problemas utilizando el lenguaje ANSI C.

Descripción del problema

Existen diversas técnicas para generar una aproximación del valor del número irracional **Pi**. En este caso utilizaremos la serie de Gregory y Leibniz.

$$\pi = 4\left(\sum_{n=1}^{\infty}\left(\frac{(-1)^{(n+1)}}{(2n-1)}\right)\right)$$

$$= \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right)$$

Procedimiento

1. Codificar una solución secuencial (sin el uso de hilos) que calcule el valor de Pi, su solución debe basarse en la serie de Gregory y Leibniz para calcular los primeros diez dígitos decimales de Pi. Para esto, utilice los primeros tomando los primeros 50,000’000,000 términos de la seria.

2. Utilice las funciones definidas en la librería **time.h** (consulte diapositivas del curso) para medir el tiempo (en milisegundos) que requiere el cálculo del valor de **Pi**. Registre el tiempo.

3. Parametrice la solución que se implemento en el paso 1.

4. Utilice hilos para repartir el trabajo de calcular el valor de **Pi**. Pruebe su solución con los siguientes casos: 2 hilos, 4 hilos, 8 hilos y 16 hilos.

5. Tomar el tiempo en milisegundos que toma el programa para calcular el valor de **Pi** en cada uno de los casos mencionados en el paso 4.

6. Registre los tiempos registrados para cada caso en la siguiente tabla:

No. de Hilos	Tiempo (milisegundos)
1	833244 ms
2	410423 ms
4	619006 ms
8	432411 ms
16	292368 ms

SOLUCIÓN DEL ALUMNO, PRUEBAS Y CONCLUSIONES

No. de Hilos	Tiempo (milisegundos)
1	833244 ms
2	410423 ms
4	619006 ms
8	432411 ms
16	292368 ms

Código fuente de la versión secuencial (sin el uso de hilos)

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
	double pi=0;
	long long i;
	clock_t start=clock();
	for(i=1;i<=50000000000;i++){
		if(((i+1)%2)==0){
			pi+=(4/(2*(float)i-1));
			//printf("%I64d\n",i);
		}
		else{
			pi-=(4/(2*(float)i-1));
			//printf("%I64d\n",i);
		}
	}
	clock_t stop=clock();
	int tiempo=1000*(stop-start)/CLOCKS_PER_SEC;
	printf("Pi=%.10lf\n",pi);
	printf("Tiempo: %d ms",tiempo);

	return 0;
}
```

Código fuente de la versión paralelizada

```
/*
 * Tarea 2 PMD
 * Hilos
*/
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 50000000000
typedef struct{
	long long st;
	long long f;
	double temp;
```

```
}Pi;

DWORD WINAPI Serie(void *);
int main(){
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);
	HANDLE h[16]={NULL};
	int opc;
	Pi pi[16];
	double final=0;
	printf("Número de hilos: ");
	scanf("%d",&opc);

	clock_t start=clock();
	for(int i=0;i<opc;i++){
		pi[i].st=(MAX/opc)*i+1;
		pi[i].f=(MAX/opc)*(i+1);
		pi[i].temp=0;
		h[i]=CreateThread(NULL, 0, Serie,(void *)&pi[i],0,NULL);
	}
	for(int i=0;i<opc;i++){
		WaitForSingleObject(h[i],INFINITE);
	}
	for(int j=0;j<opc;j++){
		final+=pi[j].temp;
	}
	clock_t stop=clock();
	int tiempo=1000*(stop-start)/CLOCKS_PER_SEC;
	printf("Pi=%.10lf\n",final);
	printf("Tiempo: %d ms",tiempo);
	return 0;
}
DWORD WINAPI Serie(void *p){
	Pi *pi=(Pi*)p;
	long long i;
	for(i=pi->st;i<=pi->f;i++){
		if(((i+1)%2)==0){
			pi->temp+=(4/(2*(float)i-1));
		}
		else{
			pi->temp-=(4/(2*(float)i-1));
		}
	}
	return 0;
}
```

Ejecución

Ejecución secuencial:

```
Pi=3.1415926536
Tiempo: 833244 ms
```

Ejecución paralelizada:

2 hilos

```
Número de hilos: 2
Pi=3.1415926536
Tiempo: 410423 ms
```

4 hilos

```
Número de hilos: 4
Pi=3.1415926536
Tiempo: 619006 ms
```

8 hilos

```
Número de hilos: 8
Pi=3.1415926536
Tiempo: 432411 ms
```

16 hilos

```
Número de hilos: 16
Pi=3.1415926536
Tiempo: 292368 ms
```

Conclusiones (obligatorio):

Dentro de esta práctica aprendí cómo se puede fragmentar un código que se ejecuta de manera secuencial para que se realice por partes utilizando los hilos, que se realizan de manera paralela, y así se la ejecución toma menos tiempo. Durante la puesta en práctica surgió la duda de si los ciclos no afectarían en la forma en que se ejecutan, es decir, que si al utilizar

un *for* para crear los hilos esto daría el mismo resultado que realizar el código de manera secuencial, pero al intentarlo se vio que no fue así y, por tanto, servía para crear un arreglo de hilos y evitar tener que crear hilo por hilo directamente en el código.

Al momento de trabajar las operaciones, lo que causó mayor problema fue la conversión de tipos de datos, ya que el contador debió utilizarse como *long long* y, al momento de usarse para la serie, debía convertirse a *float*. De igual manera, al sumar todos los datos dentro del arreglo de los resultados de las sumas por bloques, traté de hacerlo mediante una función mas no resultó como esperaba, así que puse el código dentro de la función principal y ahí no tuvo problema alguno.