
	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 1</p>

## INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
<b>ASIGNATURA:</b>	Sistema Operativos				
<b>TÍTULO DE LA PRÁCTICA:</b>					
<b>NÚMERO DE PRÁCTICA:</b>	06	<b>AÑO LECTIVO:</b>	2022-B	<b>NRO. SEMESTRE:</b>	VI (sexto)
<b>FECHA DE PRESENTACIÓN</b>	15/11/2022	<b>HORA DE PRESENTACIÓN</b>	23:59		
<b>INTEGRANTE (s):</b> Yoset Cozco Mauri				<b>NOTA:</b>	
<b>DOCENTE(s):</b> ROLANDO JESUS CARDENAS TALAVERA					

SOLUCIÓN Y RESULTADOS

### I. ACTIVIDADES

test02.c - SO - Visual Studio Code

Terminal Help
lab05\_test.c C test01.c U C test02.c U X C matriz.c U

```

lab06 > C test02.c > main()
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 struct timeval t0, t1;
5 double media = 0.0;
6 void *hilo(void *arg){
7     gettimeofday(&t1, NULL);
8     unsigned int ut1 = t1.tv_sec * 1000000 + t1.tv_usec;
9     unsigned int ut0 = t0.tv_sec * 1000000 + t0.tv_usec;
10    media += (ut1 - ut0);
11 }
12
13 int main(){
14     int i = 0;
15     pthread_t h;
16     for(i = 0; i < 100; i++){
17         gettimeofday(&t0, NULL);
18         pthread_create(&h, NULL, hilo, NULL);
19         pthread_join(h, NULL);
20     }
21     /* Tiempo medio en microsegundos*/
22     printf(" %f\n", (media / 100.0));
23 }

```



OUTPUT TERMINAL
bash - lab06

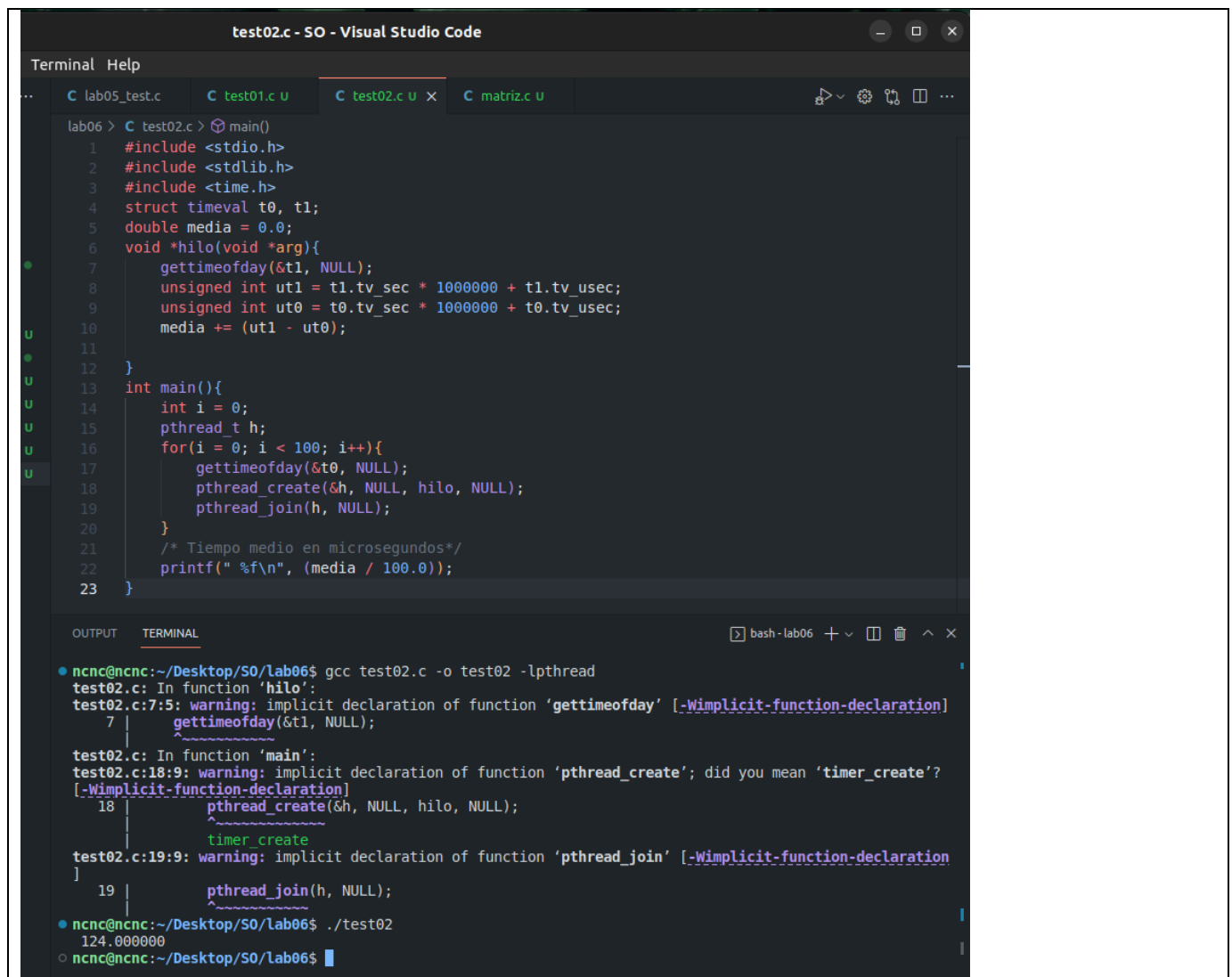
```

ncnc@ncnc:~/Desktop/SO/lab06$ gcc test01.c -o test01 -lpthread
test01.c: In function 'main':
test01.c:8:5: warning: implicit declaration of function 'gettimeofday' [-Wimplicit-function-declaration]
8     gettimeofday(&t0, NULL);
~^~^~
test01.c:10:14: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
10     id = fork();
~^~^~
ncnc@ncnc:~/Desktop/SO/lab06$ ./test01
100.300000
ncnc@ncnc:~/Desktop/SO/lab06$

```

Ln 23, Col 2   Spaces: 4   UTF-8   LF   C   Linux

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 2</p>



The screenshot shows a Visual Studio Code window titled 'test02.c - SO - Visual Studio Code'. The editor displays the source code for 'test02.c', which includes headers for `<stdio.h>`, `<stdlib.h>`, and `<time.h>`. It defines a `timeval` struct and a `hilo` function that calculates the time difference between two calls to `gettimeofday`. The `main` function creates 100 threads using `pthread_create` and joins them using `pthread_join`. The terminal output shows the compilation command `gcc test02.c -o test02 -lpthread` and several warnings about implicit function declarations for `gettimeofday`, `pthread_create`, and `pthread_join`. The final output of the program is `124.000000`.

2. Analice el código del ejemplo la sección PThreads. Describa que actividad realiza el código que se muestra. Detalle los comandos utilizados para poder ejecutar de forma correcta el código.

**Pthread\_create():** Inicia un nuevo hilo en la llamada proceso.

**Pthread\_join():** espera a que finalice el subproceso especificado por subproceso . Si ese subproceso ya terminó, `pthread_join ()` regresa inmediatamente. El subproceso especificado por subproceso debe poder unirse.

Comandos utilizados:

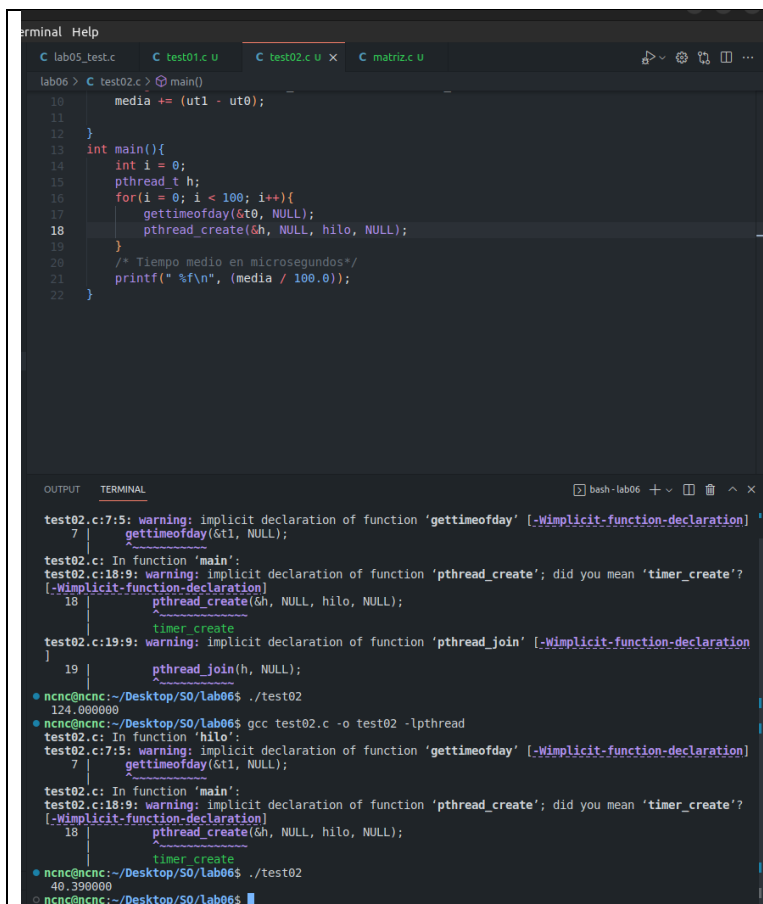
- `gcc test02.c -o test02 -lpthread`

3. Explique a que se debe el orden de ejecución del proceso principal y del Thread creado, ¿Se ejecutan en paralelo o de forma secuencial?

Secuencial, dado que hay solo un hilo h. sobre el que se ejecuta método `create` y `join`.

4. Del código de la sección PThreads, que ocurre si se remueve la sentencia `pthread_exit(0)`. Elabore un ejemplo de código que demuestre el comportamiento del código sin esta sentencia.

5. De igual manera, que involucra remover la sentencia `pthread_join(hilo, NULL)`. ¿Existen cambios significativos en el comportamiento de los códigos? Elabore un ejemplo en código.



```
terminal Help
C lab05_test.c C test01.c U C test02.c U X C matriz.c U
lab06 > C test02.c > main()
10 media += (ut1 - ut0);
11
12 }
13 int main(){
14 int i = 0;
15 pthread_t h;
16 for(i = 0; i < 100; i++){
17 gettimeofday(&t0, NULL);
18 pthread_create(&h, NULL, hilo, NULL);
19 }
20 /* Tiempo medio en microsegundos*/
21 printf(" %f\n", (media / 100.0));
22 }

OUTPUT TERMINAL
test02.c:7:5: warning: implicit declaration of function 'gettimeofday' [-Wimplicit-function-declaration]
7     gettimeofday(&t1, NULL);
test02.c: In function 'main':
test02.c:18:9: warning: implicit declaration of function 'pthread_create'; did you mean 'timer_create'? [-Wimplicit-function-declaration]
18     pthread_create(&h, NULL, hilo, NULL);
test02.c:19:9: warning: implicit declaration of function 'pthread_join' [-Wimplicit-function-declaration]
19     pthread_join(h, NULL);
ncnc@ncnc:~/Desktop/S0/Lab06$ ./test02
124.000000
ncnc@ncnc:~/Desktop/S0/Lab06$ gcc test02.c -o test02 -lpthread
test02.c: In function 'hilo':
test02.c:7:5: warning: implicit declaration of function 'gettimeofday' [-Wimplicit-function-declaration]
7     gettimeofday(&t1, NULL);
test02.c: In function 'main':
test02.c:18:9: warning: implicit declaration of function 'pthread_create'; did you mean 'timer_create'? [-Wimplicit-function-declaration]
18     pthread_create(&h, NULL, hilo, NULL);
ncnc@ncnc:~/Desktop/S0/Lab06$ ./test02
40.390000
ncnc@ncnc:~/Desktop/S0/Lab06$
```

Reduce el tiempo de ejecución.

6. En el siguiente ejercicio se intenta medir el tiempo de creación de 100 procesos (pesados) a partir un proceso padre, y el tiempo de creación de 100 hilos. Observe los resultados de ambos programas. Haga pruebas con un valor mayor de procesos y de hilos. ¿Qué conclusiones le merece los resultados observados?

## II. SOLUCIÓN DE EJERCICIOS/PROBLEMAS

### EJERCICIOS PROPUESTOS



1. Modifique el programa *Ejercicio\_Matriz* de tal forma que, un hilo se encargue de multiplicar cada fila de la matriz por un escalar cualquiera N.

```
C matriz.c u x
lab06 > C matriz.c > matrizporescalar(void *)
1
2
3
4 #include <unistd.h>
5 #include <pthread.h>
6
7 struct parametros {
8     int id;
9     float escalar;
10    float matriz[3][3];
11 };
12
13 void init (float m [3][3]) {
14     int i, j;
15     for (i = 0; i < 3; i++) {
16         for (j = 0; j < 3; j++) {
17             m[i][j] = random()*100;
18         }
19     }
20 }
21 void *matrizporescalar(void *arg){
22     struct parametros *p;
23     int i;
24     int j;
25     p = (struct parametros *) arg;
26     for (i = 0; i < 3; i++) {
27         printf("Hilo %d multiplicando fila %d\n", p->id, i);
28         for (j = 0; j < 3; j++) {
29             p->matriz[i][j] = p->matriz[i][j] * p->escalar;
30             sleep(1);
31         }
32     }
33 }
34 int main(int argc, char *argv []){
35     pthread_t h1;
36     struct parametros p1;
37     p1.id = 1;
38     p1.escalar = 5.0;
39
40     init(p1.matriz);
41     pthread_create(&h1, NULL, matrizporescalar, (void *)&p1);
42     pthread_join(h1, NULL);
43     printf("Fin \n");
44 }
45 }
```

**OUTPUT** **TERMINAL**

```
● ncnc@ncnc:~/Desktop/S0/lab06$ gcc matriz.c -o matriz -lpthread
● ncnc@ncnc:~/Desktop/S0/lab06$ ./matriz
Hilo 1 multiplicando fila 0
Hilo 1 multiplicando fila 1
Hilo 1 multiplicando fila 2
Fin
○ ncnc@ncnc:~/Desktop/S0/lab06$
```

2. Modifique el programa *Ejercicio\_Matriz* de modo que a través de otro hilo se muestre el contenido de la matriz.
3. Elabore un programa con un mínimo de 4 Threads, que realice la suma de una lista de números (esta estructura contendrá más de 5000 elementos generados de forma aleatoria). Elabore un programa que realice el proceso de forma secuencial para validar los resultados obtenidos.

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 5</p>

Mida los tiempos requeridos por ambos programas (con varios hilos/un solo hilo).  
¿A qué conclusiones puede usted llegar después de esta experiencia?

### III. SOLUCIÓN DEL CUESTIONARIO

1. ¿Qué diferencia existe entre crear un proceso con PThread y Fork?

Fork:

- Propósito crear un nuevo proceso, que se convierte en proceso secundario del proceso que lo llama.
- Ambos procesos ejecutaran la siguiente instrucción después de la llamada al sistema fork().

Pthread:

- Propósito crear un nuevo hilo en el programa al que se le da el mismo proceso de la persona que lo llama.
- Los subprocesos dentro del mismo proceso pueden comunicarse utilizando la memoria compartida.

2. ¿Por qué no es posible compartir una variable declarada dentro de la función *main( )* entre varios PThreads?



Porque los hilos suelen compartir datos estáticos, espacios de dirección, segmentos de códigos, dado que cada subproceso tiene su propia copia de las variables locales.

3. Si existen 2 PThreads que modifican una variable global (uno realiza la operación  $x += 1$  y el otro  $x -= 1$ , repitiendo 100 veces cada operación ) y se muestra cada operación en pantalla. ¿Se llega a obtener siempre el mismo resultado o este cambia en cada ejecución? Escriba un programa y realice varias pruebas que sustenten sus afirmaciones.

### IV. CONCLUSIONES

### RETROALIMENTACIÓN GENERAL

### REFERENCIAS Y BIBLIOGRAFÍA

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 6</p>