

PRÁCTICA 4 DE SISTEMAS OPERATIVOS

TEMA: Creación de hilos.

Nombre: Fernando Eliceo Huilca Villagómez

Carrera: Ingeniería de Software

Grupo: GR1SW

Fecha: 17/ 07 / 2024

Índice de Contenidos

1. OBJETIVOS	2
2. INFORME.....	2
Crear 4 hilos y cada uno que imprima un mensaje diferente.....	2
Crear 5 hilos y que cada uno pase un parámetro numérico a una función que devolverá la suma de este número con un valor entero declarado dentro de la función.	3
Utilizando estructuras muestre el resultado del producto de un escalar por valores dentro de una matriz.....	4
Escribir código para determinar el resultado del tiempo de ejecución de 1 millón, 2 millones, y 3 millones de hilos. Tomar los tiempos en microsegundos.....	4
3. CONCLUSIONES Y RECOMENDACIONES.....	6
4. BIBLIOGRAFÍA	7

Índice de ilustraciones

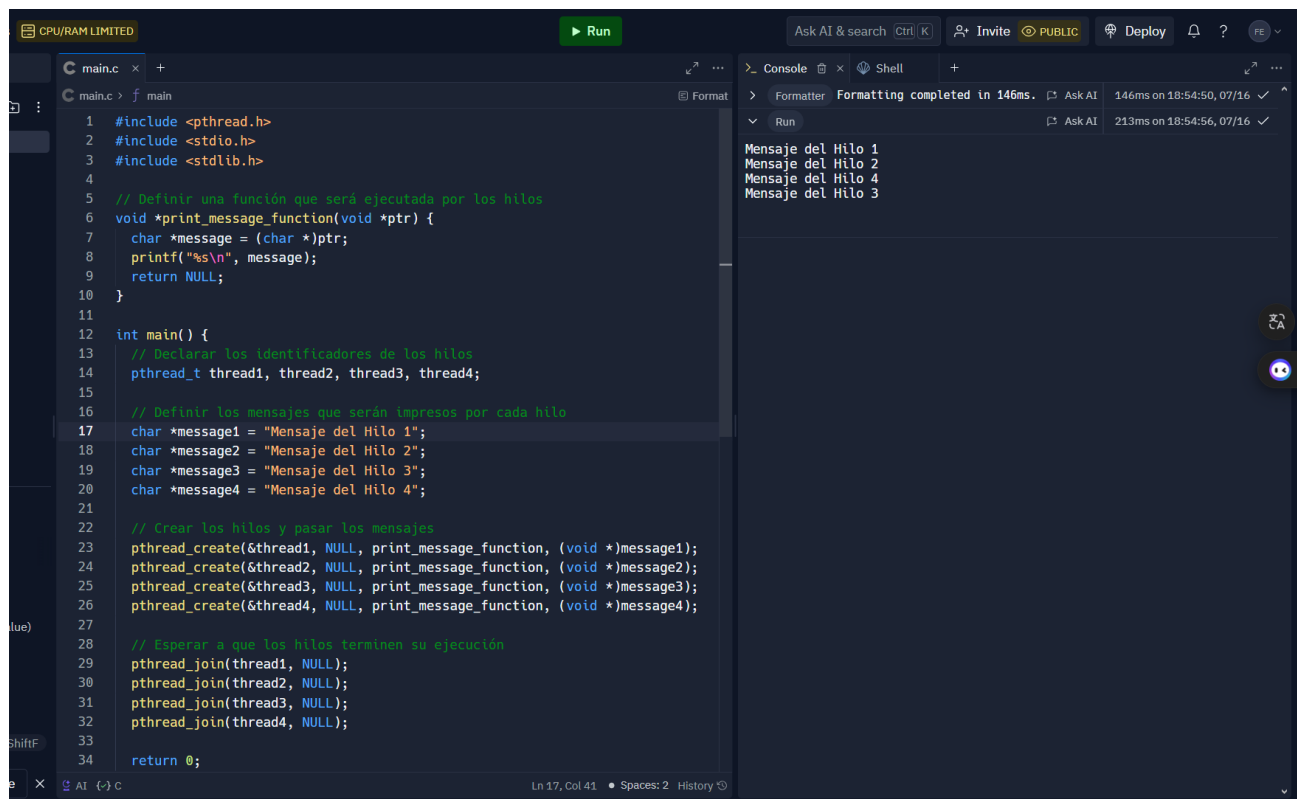
Ilustración 1 Creación de 4 hilos e imprimir un mensaje diferente	2
Ilustración 2 Cinco hilos que cada uno pasa un parámetro numérico y devuelve la suma	3
Ilustración 3 Resultado de la ejecución de un escalar por valores dentro de una matriz.....	4
Ilustración 4 Ejecución fallida de 1, 2 3, millones de hilos en mycompiler.io	5
Ilustración 5 Ejecución fallida de 1, 2 3, millones de hilos en replit	5
Ilustración 6 Resultado exitoso del ejercicio 4	6

1. OBJETIVOS

- 1.1. Familiarizar al estudiante con el uso de las funciones **pthread**.
- 1.2. Realizar varias actividades de creación de hilos.
- 1.3. Comprender el manejo de estructuras y el paso de parámetros en funciones de hilos.
- 1.4. Evaluar el desempeño del sistema al crear y unir un gran número de hilos.
- 1.5. Analizar los tiempos de ejecución en microsegundos para diferentes cantidades de hilos.

2. INFORME

Crear 4 hilos y cada uno que imprima un mensaje diferente.



```
1 #include <pthread.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 // Definir una función que será ejecutada por los hilos
6 void *print_message_function(void *ptr) {
7     char *message = (char *)ptr;
8     printf("%s\n", message);
9     return NULL;
10 }
11
12 int main() {
13     // Declarar los identificadores de los hilos
14     pthread_t thread1, thread2, thread3, thread4;
15
16     // Definir los mensajes que serán impresos por cada hilo
17     char *message1 = "Mensaje del Hilo 1";
18     char *message2 = "Mensaje del Hilo 2";
19     char *message3 = "Mensaje del Hilo 3";
20     char *message4 = "Mensaje del Hilo 4";
21
22     // Crear los hilos y pasar los mensajes
23     pthread_create(&thread1, NULL, print_message_function, (void *)message1);
24     pthread_create(&thread2, NULL, print_message_function, (void *)message2);
25     pthread_create(&thread3, NULL, print_message_function, (void *)message3);
26     pthread_create(&thread4, NULL, print_message_function, (void *)message4);
27
28     // Esperar a que los hilos terminen su ejecución
29     pthread_join(thread1, NULL);
30     pthread_join(thread2, NULL);
31     pthread_join(thread3, NULL);
32     pthread_join(thread4, NULL);
33
34     return 0;
35 }
```

Console output:

```
Mensaje del Hilo 1
Mensaje del Hilo 2
Mensaje del Hilo 4
Mensaje del Hilo 3
```

Ilustración 1 Creación de 4 hilos e imprimir un mensaje diferente

Inclusión de Bibliotecas: Se incluyen las bibliotecas estándar de C (stdio.h, stdlib.h) y la de pthreads (pthread.h).

Definición de la Función del Hilo: La función `print_message_function` toma un puntero genérico (`void*`), lo convierte a un puntero a `char` y luego imprime el mensaje.

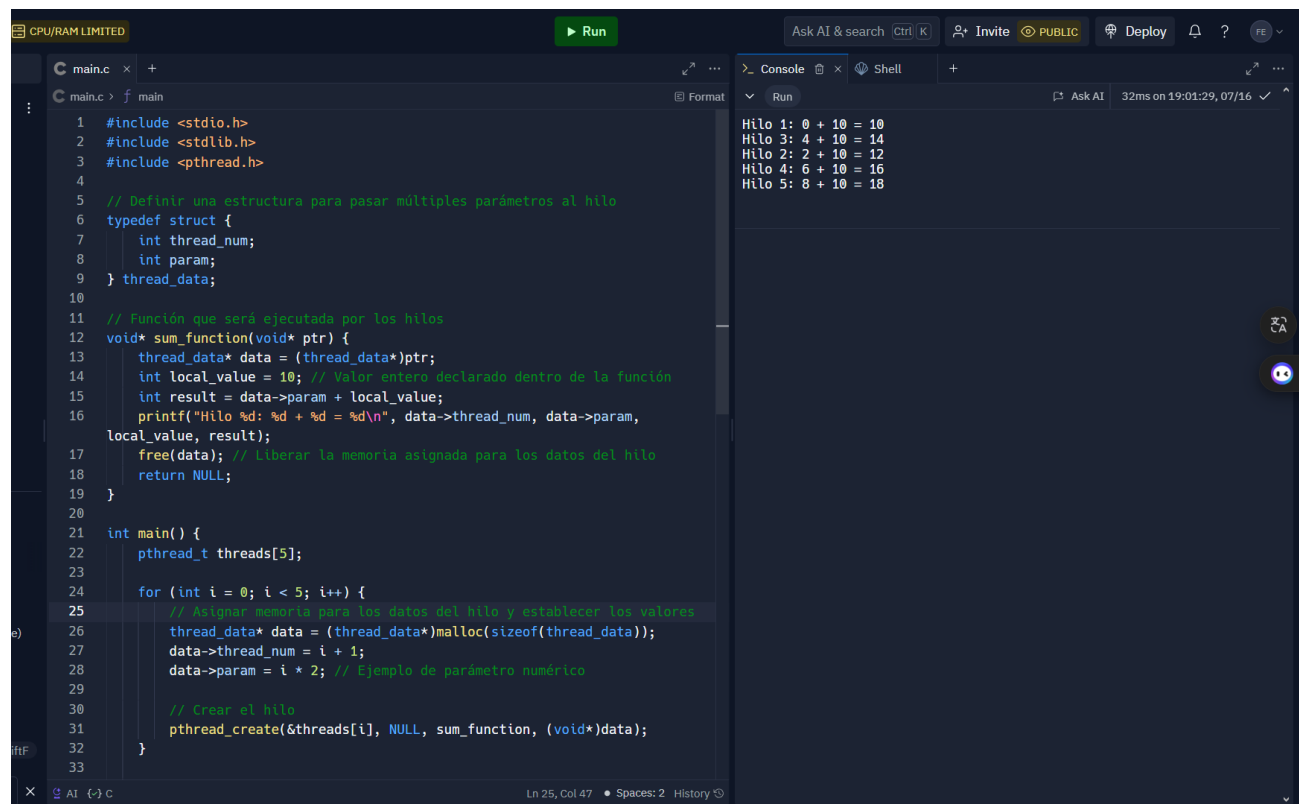
Declaración de los Identificadores de los Hilos: Se declaran cuatro variables de tipo `pthread_t` para los hilos.

Definición de los Mensajes: Se crean cuatro cadenas de caracteres con los mensajes que cada hilo imprimirá.

Creación de los Hilos: Se crean los hilos con `pthread_create`, pasando la función `print_message_function` y los mensajes respectivos.

Esperar a que los Hilos Terminen: Se utiliza `pthread_join` para esperar a que los hilos terminen su ejecución antes de que el programa principal continúe.

Crear 5 hilos y que cada uno pase un parámetro numérico a una función que devolverá la suma de este número con un valor entero declarado dentro de la función.

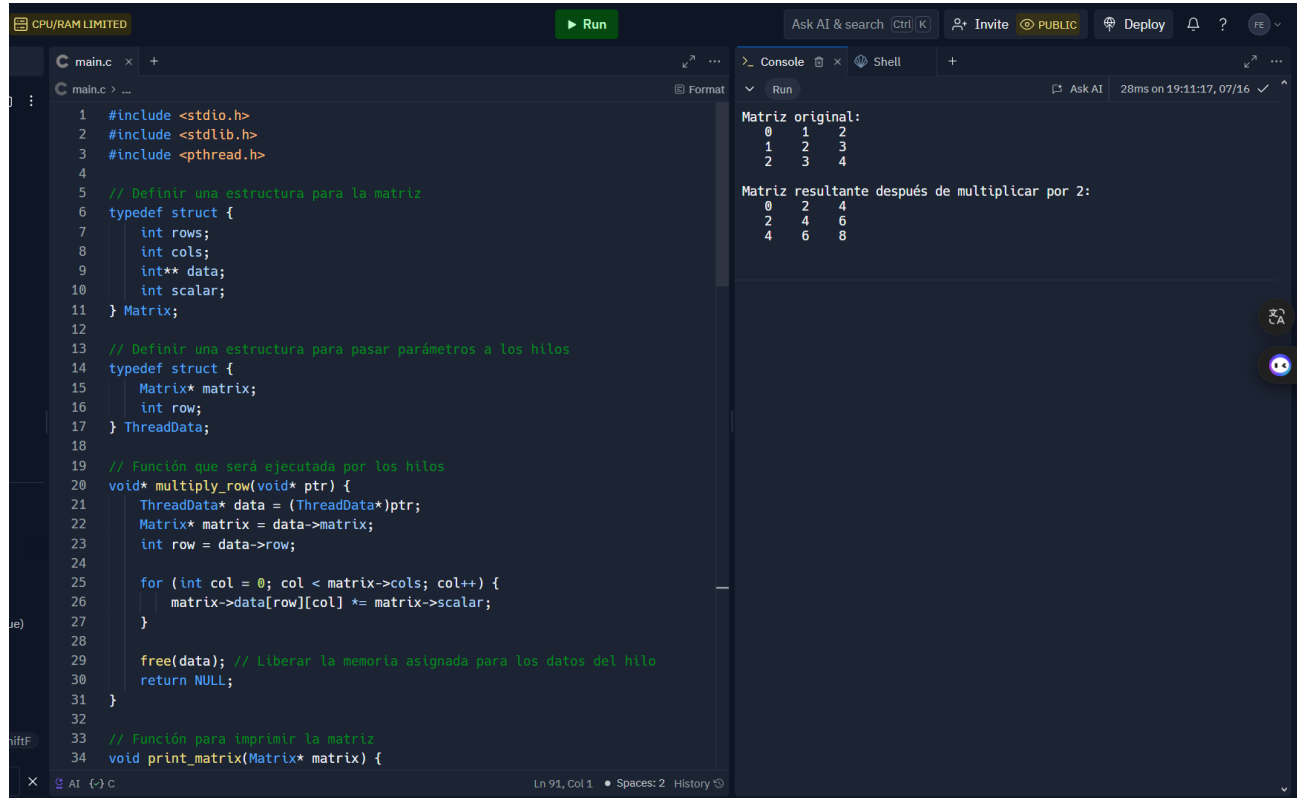


```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4
5 // Definir una estructura para pasar múltiples parámetros al hilo
6 typedef struct {
7     int thread_num;
8     int param;
9 } thread_data;
10
11 // Función que será ejecutada por los hilos
12 void* sum_function(void* ptr) {
13     thread_data* data = (thread_data*)ptr;
14     int local_value = 10; // Valor entero declarado dentro de la función
15     int result = data->param + local_value;
16     printf("Hilo %d: %d + %d = %d\n", data->thread_num, data->param,
17         local_value, result);
18     free(data); // Liberar la memoria asignada para los datos del hilo
19     return NULL;
20 }
21
22 int main() {
23     pthread_t threads[5];
24
25     for (int i = 0; i < 5; i++) {
26         // Asignar memoria para los datos del hilo y establecer los valores
27         thread_data* data = (thread_data*)malloc(sizeof(thread_data));
28         data->thread_num = i + 1;
29         data->param = i * 2; // Ejemplo de parámetro numérico
30
31         // Crear el hilo
32         pthread_create(&threads[i], NULL, sum_function, (void*)data);
33     }
34 }
```

Hilo 1: 0 + 10 = 10
Hilo 3: 4 + 10 = 14
Hilo 2: 2 + 10 = 12
Hilo 4: 6 + 10 = 16
Hilo 5: 8 + 10 = 18

Ilustración 2 Cinco hilos que cada uno pasa un parámetro numérico y devuelve la suma

0055tilizando estructuras muestre el resultado del producto de un escalar por valores dentro de una matriz.



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4
5 // Definir una estructura para la matriz
6 typedef struct {
7     int rows;
8     int cols;
9     int** data;
10    int scalar;
11 } Matrix;
12
13 // Definir una estructura para pasar parámetros a los hilos
14 typedef struct {
15     Matrix* matrix;
16     int row;
17 } ThreadData;
18
19 // Función que será ejecutada por los hilos
20 void* multiply_row(void* ptr) {
21     ThreadData* data = (ThreadData*)ptr;
22     Matrix* matrix = data->matrix;
23     int row = data->row;
24
25     for (int col = 0; col < matrix->cols; col++) {
26         matrix->data[row][col] *= matrix->scalar;
27     }
28
29     free(data); // Liberar la memoria asignada para los datos del hilo
30     return NULL;
31 }
32
33 // Función para imprimir la matriz
34 void print_matrix(Matrix* matrix) {
```

Matriz original:

0	1	2
1	2	3
2	3	4

Matriz resultante después de multiplicar por 2:

0	2	4
2	4	6
4	6	8

Ilustración 3 Resultado de la ejecución de un escalar por valores dentro de una matriz

Escribir código para determinar el resultado del tiempo de ejecución de 1 millón, 2 millones, y 3 millones de hilos. Tomar los tiempos en microsegundos.



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <sys/time.h>
5
6 // Función que será ejecutada por los hilos (sin operación)
7 void* thread_function(void* arg) {
8     return NULL;
9 }
10
11 // Función para medir el tiempo de ejecución de la creación de hilos
12 void measure_thread_creation_time(int num_threads) {
13     struct timeval start, end;
14     pthread_t* threads = (pthread_t*)malloc(num_threads * sizeof(pthread_t));
15
16     if (threads == NULL) {
17         fprintf(stderr, "Error al asignar memoria para los hilos\n");
18         exit(EXIT_FAILURE);
19     }
20
21     // Tomar el tiempo de inicio
22     gettimeofday(&start, NULL);
23
24     // Crear los hilos
25     for (int i = 0; i < num_threads; i++) {
26         if (pthread_create(&threads[i], NULL, thread_function, NULL) != 0) {
27             fprintf(stderr, "Error al crear el hilo %d\n", i);
28             exit(EXIT_FAILURE);
29         }
30     }
31 }
```

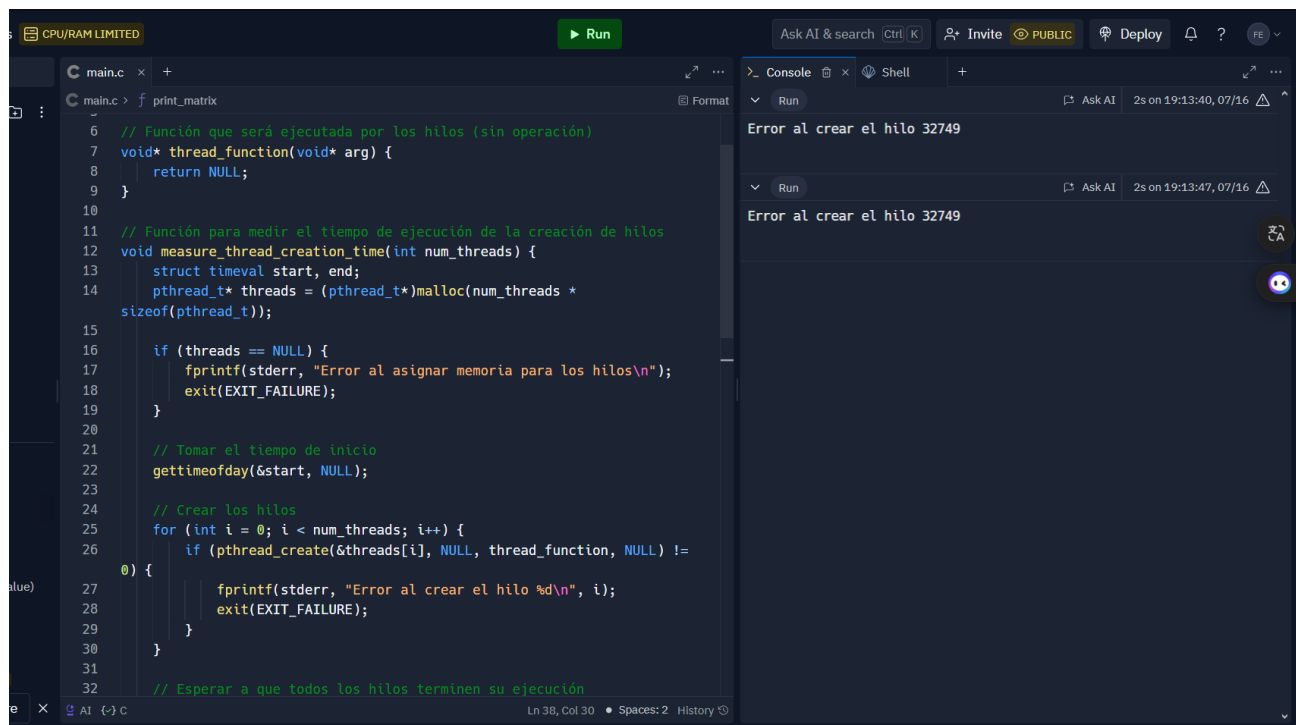
Entrada del programa

Salida del programa

Error al crear el hilo 14921

[Execution complete with exit code 1]

Ilustración 4 Ejecución fallida de 1, 2 3, millones de hilos en mycompiler.io

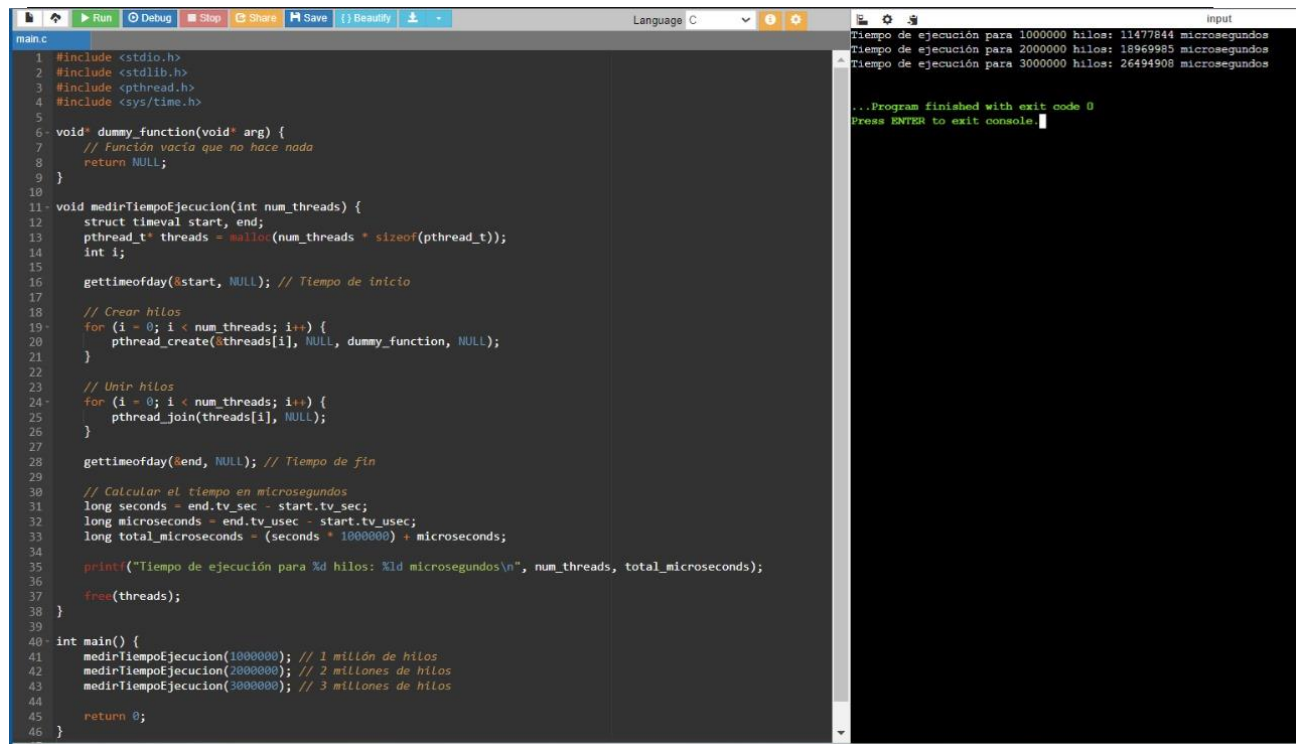


```
6 // Función que será ejecutada por los hilos (sin operación)
7 void* thread_function(void* arg) {
8     return NULL;
9 }
10
11 // Función para medir el tiempo de ejecución de la creación de hilos
12 void measure_thread_creation_time(int num_threads) {
13     struct timeval start, end;
14     pthread_t* threads = (pthread_t*)malloc(num_threads *
15     sizeof(pthread_t));
16
17     if (threads == NULL) {
18         fprintf(stderr, "Error al asignar memoria para los hilos\n");
19         exit(EXIT_FAILURE);
20     }
21
22     // Tomar el tiempo de inicio
23     gettimeofday(&start, NULL);
24
25     // Crear los hilos
26     for (int i = 0; i < num_threads; i++) {
27         if (pthread_create(&threads[i], NULL, thread_function, NULL) !=
28         0) {
29             fprintf(stderr, "Error al crear el hilo %d\n", i);
30             exit(EXIT_FAILURE);
31         }
32     }
33
34     // Esperar a que todos los hilos terminen su ejecución
35 }
```

Error al crear el hilo 32749

Error al crear el hilo 32749

Ilustración 5 Ejecución fallida de 1, 2 3, millones de hilos en repl.it



```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <sys/time.h>
5
6 void* dummy_function(void* arg) {
7     // función vacía que no hace nada
8     return NULL;
9 }
10
11 void medirTiempoEjecucion(int num_threads) {
12     struct timeval start, end;
13     pthread_t* threads = malloc(num_threads * sizeof(pthread_t));
14     int i;
15
16     gettimeofday(&start, NULL); // Tiempo de inicio
17
18     // Crear hilos
19     for (i = 0; i < num_threads; i++) {
20         pthread_create(&threads[i], NULL, dummy_function, NULL);
21     }
22
23     // Unir hilos
24     for (i = 0; i < num_threads; i++) {
25         pthread_join(threads[i], NULL);
26     }
27
28     gettimeofday(&end, NULL); // Tiempo de fin
29
30     // Calcular el tiempo en microsegundos
31     long seconds = end.tv_sec - start.tv_sec;
32     long microseconds = end.tv_usec - start.tv_usec;
33     long total_microseconds = (seconds * 1000000) + microseconds;
34
35     printf("Tiempo de ejecución para %d hilos: %ld microsegundos\n", num_threads, total_microseconds);
36     free(threads);
37 }
38
39 int main() {
40     medirTiempoEjecucion(1000000); // 1 millón de hilos
41     medirTiempoEjecucion(2000000); // 2 millones de hilos
42     medirTiempoEjecucion(3000000); // 3 millones de hilos
43
44     return 0;
45 }
```

input

Tiempo de ejecución para 1000000 hilos: 11477844 microsegundos
Tiempo de ejecución para 2000000 hilos: 18969985 microsegundos
Tiempo de ejecución para 3000000 hilos: 26494908 microsegundos

...Program finished with exit code 0
Press ENTER to exit console.

Ilustración 6 Resultado exitoso del ejercicio 4

3. CONCLUSIONES Y RECOMENDACIONES

Conclusiones:

- El uso de la biblioteca pthread en C permite la creación y manejo de hilos, lo cual es fundamental para tareas concurrentes y paralelas en programación.
- La implementación y manejo correcto de estructuras en C facilita el paso de múltiples parámetros a las funciones de los hilos.
- La creación de millones de hilos para medir el tiempo de ejecución puede ser impracticable debido a las limitaciones de recursos del sistema. En la práctica, la máquina puede no tener suficiente memoria o capacidad de procesamiento para manejar tantas operaciones concurrentes simultáneamente.

Recomendaciones:

- No usar compiladores virtuales: Para tareas intensivas en recursos, como la creación de millones de hilos, se recomienda no utilizar compiladores virtuales o en línea. Es preferible usar un IDE completo y robusto instalado localmente en el sistema, como Visual Studio Code, Eclipse, o cualquier otro IDE de tu preferencia.
- Optimización de recursos: En aplicaciones reales, considera el uso de un thread pool o un modelo de concurrencia más eficiente, en lugar de crear un número excesivo de hilos, para optimizar el uso de recursos del sistema.
- Monitoreo y manejo de errores: Implementa siempre monitoreo y manejo de errores adecuado al crear y unir hilos para evitar fallos inesperados en el programa.

4. BIBLIOGRAFÍA

[1] GNU C Library Documentation, "The GNU C Library," [Online]. Available: <https://www.gnu.org/software/libc/manual/>. [Accessed: 17-Jul-2024].

[2] Pthreads Tutorial, "POSIX Threads Programming," Lawrence Livermore National Laboratory, [Online]. Available: <https://computing.llnl.gov/tutorials/pthreads/>. [Accessed: 17-Jul-2024].

[3] The Open Group Base Specifications Issue 7, "The Open Group Base Specifications Issue 7," [Online]. Available: <https://pubs.opengroup.org/onlinepubs/9699919799/>. [Accessed: 17-Jul-2024].