

# Universidad de Guadalajara

---

## Ingeniería Informática



### Hands-on 6 & 7: Thread Programming

**SEMINARIO DE SOLUCION DE PROBLEMAS DE USO, ADAPTACION, EXPLOTACION  
DE SISTEMAS OPERATIVOS - 103857 - D03**

**Alumno: Ramirez Gomez Kevin Ramses**

**Código: 218218194**

**Profesor: Jose Antonio Aviña Mendez**

**Fecha: 14/11/2024**



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De  
Sistemas Operativos - 103857 - D03

## Índice

Desarrollo.....	3
Hand on 6.....	3
Mi analisis del programa y comprension de los hilos. ....	5
Hand on 7.....	5
Mi análisis del programa.....	7



## DESARROLLO

### HAND ON 6

Necesitaremos crear un archivo que lo llamaremos threads.c y añadimos el código.

The screenshot shows a virtual machine window titled 'MaquinaUbuntu [Running] - Oracle VM VirtualBox'. Inside the VM, the desktop environment is Ubuntu. A terminal window is open, displaying the following C code in a nano editor:

```
GNU nano 7.2 threads.c *
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

// Función que ejecutará cada hilo
void* print_message_function(void* thread_id) {
    long tid = (long)thread_id;
    printf("Hilo %ld está en ejecución.\n", tid);
    // Simula trabajo con sleep
    sleep(2);
    printf("Hilo %ld ha terminado.\n", tid);
    pthread_exit(NULL);
}

int main() {
    const int NUM_THREADS = 5;
    pthread_t threads[NUM_THREADS];
    int rc;
    long t;

    for (t = 0; t < NUM_THREADS; t++) {
        rc = pthread_create(&threads[t], NULL, print_message_function, (void*)t);
        if (rc) {
            printf("Error: pthread_create() returned %d\n", rc);
            exit(-1);
        }
    }

    for (t = 0; t < NUM_THREADS; t++) {
        pthread_join(threads[t], NULL);
    }

    printf("Todos los hilos han terminado.\n");
    return 0;
}
```



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De  
Sistemas Operativos - 103857 - D03

Ahora ejecutamos el siguiente comando para crear el archivo de ejecucion.

```
KevinRamirezGomez@MaquinaUbuntu:~$ nano threads.c
KevinRamirezGomez@MaquinaUbuntu:~$ gcc -pthread threads.c -o threads
KevinRamirezGomez@MaquinaUbuntu:~$ ls
AUTO-NV.sh  processinfo.c  segment_size2  snap
create.sh   prueba1.txt    segment_size2.c  Templates
Desktop     prueba2.txt    segment_size3.c  Test
Documents   pruebaawk.txt  segment_size3.c  threads
Downloads   pruebaagrep.txt segment_size4    threads.c
helloworld.sh prueba.txt      segment_size4.c  Videos
Music       Public         segment_size5    'VirtualBox VHS'
Pictures    segment_size1  segment_size5.c
processinfo segment_size1.c services-menu.sh
KevinRamirezGomez@MaquinaUbuntu:~$
```

Ahora ejecutamos el archivo de ejecución y vemos lo que nos da de resultado.

```
KevinRamirezGomez@MaquinaUbuntu:~$ ./threads
Creando el hilo 0
Creando el hilo 1
Creando el hilo 2
Creando el hilo 3
Creando el hilo 4
Hilo 0 está en ejecución.
Hilo 1 está en ejecución.
Hilo 4 está en ejecución.
Hilo 3 está en ejecución.
Hilo 2 está en ejecución.
Hilo 0 ha terminado.
Hilo 3 ha terminado.
Hilo 2 ha terminado.
Hilo 1 ha terminado.
Hilo 4 ha terminado.
Todos los hilos han terminado.
KevinRamirezGomez@MaquinaUbuntu:~$
```



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De Sistemas Operativos - 103857 - D03

## Mi analisis del programa y comprension de los hilos.

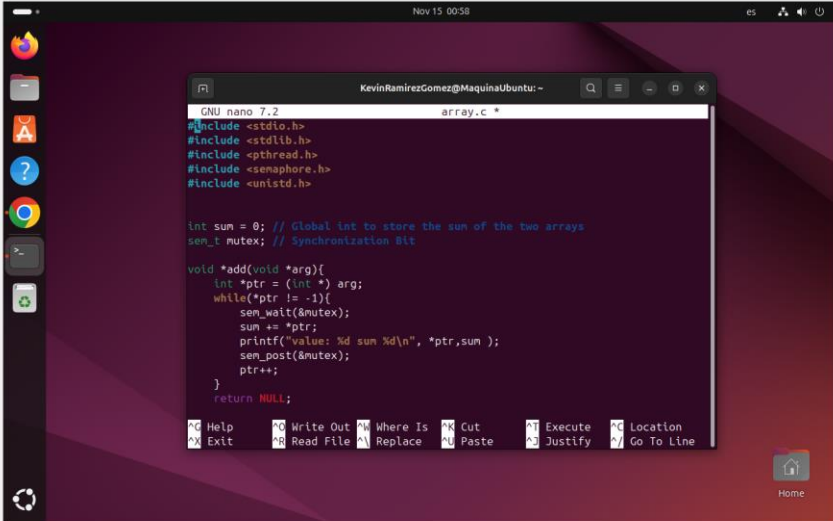
Al trabajar en este programa en C para crear y administrar hilos, me di cuenta de lo esencial que es entender la programación concurrente y paralela. La posibilidad de dividir un proceso en múltiples tareas independientes y hacer que se ejecuten en paralelo me hizo reflexionar sobre la eficiencia que se puede lograr en aplicaciones que requieren un alto rendimiento. Saber que con `pthread_create` puedo iniciar hilos y con `pthread_join` asegurarme de que todos terminen antes de que el programa avance, me dio una visión más clara de cómo gestionar los estados de vida de un hilo.

Me sorprendió lo importante que es planificar cuidadosamente el ciclo de vida de los hilos, ya que manejar su creación, ejecución y finalización de manera correcta es fundamental para evitar problemas como condiciones de carrera o errores de sincronización. Es un recordatorio de que aunque los hilos ofrecen grandes beneficios, también pueden traer una complejidad significativa si no se gestionan adecuadamente.

También me resultó interesante ver cómo los hilos pueden ejecutar la misma función de manera simultánea, lo que me abrió los ojos a las posibilidades de crear aplicaciones más complejas y eficientes, como servidores que atienden múltiples solicitudes al mismo tiempo o programas que procesan datos en paralelo.

## HAND ON 7

Creemos un archivo llamado `array.c` con el código que viene en la página.



```
GNU nano 7.2 array.c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

int sum = 0; // Global int to store the sum of the two arrays
sem_t mutex; // Synchronization #1

void *add(void *arg){
    int *ptr = (int *) arg;
    while(*ptr != -1){
        sem_wait(&mutex);
        sum += *ptr;
        printf("value: %d sum %d\n", *ptr, sum);
        sem_post(&mutex);
        ptr++;
    }
    return NULL;
}
```



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De  
Sistemas Operativos - 103857 - D03

Ahora hacemos el ejecutable del archivo.

```
KevinRamirezGomez@MaquinaUbuntu:~$ nano array.c
KevinRamirezGomez@MaquinaUbuntu:~$ gcc -pthread array.c -o array
KevinRamirezGomez@MaquinaUbuntu:~$ ls
array      Pictures      segment_size1  segment_size5.c
array.c    processinfo  segment_size1.c services-menu.sh
AUTO-MV.sh processinfo.c segment_size2.c snap
create.sh  prueba1.txt  segment_size2.c Templates
Desktop    prueba2.txt  segment_size3  Test
Documents  pruebaawk.txt segment_size3.c threads
Downloads  pruebagrep.txt segment_size4  threads.c
helloworld.sh prueba.txt  segment_size4.c Videos
Music      Public      segment_size5  'VirtualBox VMs'
```

Ejecutamos el archivo ejecutable.

```
KevinRamirezGomez@MaquinaUbuntu:~$ nano array.c
KevinRamirezGomez@MaquinaUbuntu:~$ gcc -pthread array.c -o array
KevinRamirezGomez@MaquinaUbuntu:~$ ls
array      Pictures      segment_size1  segment_size5.c
array.c    processinfo  segment_size1.c services-menu.sh
AUTO-MV.sh processinfo.c segment_size2.c snap
create.sh  prueba1.txt  segment_size2.c Templates
Desktop    prueba2.txt  segment_size3  Test
Documents  pruebaawk.txt segment_size3.c threads
Downloads  pruebagrep.txt segment_size4  threads.c
helloworld.sh prueba.txt  segment_size4.c Videos
Music      Public      segment_size5  'VirtualBox VMs'
```

```
KevinRamirezGomez@MaquinaUbuntu:~$ ./array
value: 4 sum 4
value: 2 sum 6
value: 6 sum 12
value: 1 sum 13
value: 2 sum 15
value: 3 sum 18
Total: 18
KevinRamirezGomez@MaquinaUbuntu:~$
```



# Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Seminario De Solucion De Problemas De Uso, Adaptacion, Explotacion De  
Sistemas Operativos - 103857 - D03

## **Mi análisis del programa.**

Este programa me permitió comprender cómo sumar dos arreglos de números enteros utilizando la interacción entre dos hilos. Cada hilo se encarga de sumar los elementos de un arreglo, y mediante el uso de un semáforo, se garantiza que la variable global sum se actualice de manera segura, evitando condiciones de carrera. El semáforo actúa como un mecanismo de sincronización para asegurar que solo un hilo modifique sum a la vez. A través de este enfoque, pude ver cómo la paralelización puede hacer que tareas como la suma de arreglos sean más eficientes, distribuyendo el trabajo entre hilos y luego combinando los resultados de manera controlada. La interacción entre los hilos, junto con la sincronización adecuada, asegura que la suma de los elementos de ambos arreglos se realice correctamente, manteniendo la coherencia de los datos.