# Integrantes:

Oscar Orozco: 15584103
Gustavo Álvarez: 15584101

# Principles of Parallel Programming - Chapter One

# Threads in C

## A gentle introduction

The first example is `thread_101_1.c`. This program exhibits how interact with pthread methods such as `pthread_create` and `pthread_join`.

In order to compile the program run the following command `gcc -pthread thread_101_1.c -o thread_101_1`

### Assignment

Remove the instructions related to `pthread_join` and explain:

- What is the program's behavior

Antes de remover el `pthread_join` siempre se obtenía como parte de la respuesta en la ejecución la confirmación de que el incremento de las dos variables terminaba. Ahora, sólo el incremento de “y” finaliza y el valor de x no ha sido modificado.

- If it is unexpected, what did you think that happen?

Analizando el código es claro que el incremento de “y” finaliza, dado que hace parte de la declaración principal, es el incremento de “x” el que está separado en un hilo. Es decir que el programa termina su ejecución sin que haya terminado la ejecución del hilo que incrementa “x”, es más ni siquiera se entra en la primera iteración del incremento. Esto puede confirmarse si se incrementa considerablemente las

iteraciones del for para el incremento de “y”, de tal manera que el incremento de “x” logra terminar antes.

Try the program several times.

- All the executions were equal?

En principio con un incremento de 100 veces por iteración no se ve ninguna diferencia. Pero si se aumentan los ciclos de los incrementos, por ejemplo a 10000, se ve que en cada ejecución resulta un valor diferente de x, casi siempre por debajo de 80000. Esto puede deberse a que cada vez que ejecuto el programa, el estado del pc cambia, es decir, su capacidad disponible de procesamiento principalmente.

## Another example of using threads

In chapter one of the book "Principles of Parallel Programming", a trivial example using threads is presented. The problem is to count the number of times that the number "3" is present in a given array.

Example if the input array is `[10, 3, 15, 4, 10, 3, 2, -1]` the program should return `2`.

In this directory you find different C programs to present the code that should help you on understanding the section "Parallelism Using Multiple Instruction Streams" in the aforementioned chapter. As follows a description of every program is given.

# Important Note

To compile all programs in this directory run

```
make clean && make all
```

**Note:** It is very important to read chapter one of "Principles of Parallel Programming" before to continue with this reading.

**Note:** These programs were run in a computer having 8 GB of RAM

# 3s-00.c

This program presents a sequential solution of the problem stated in chapter one. The code has two principal parts:

- Initialize and populate a large vector with random numbers.
- Count in the initialized vector the number of 3s in it.

## Assignment

Add the proper code sentences that allow you to determine how much time the program spends during:

- Vector initialization.

Utiliza alrededor de 2 segundo para la inicialización. Algunos tiempos obtenidos son: 1956 ms, 2094 ms y 1965 ms.

- Counting the number of 3s in a given vector.

Los tiempos para este proceso toman entre 283 y 286 ms.

NOTA: dado que los ejemplos están diseñados para un equipo con 8G de RAM, se tuvo que bajar el tamaño del vector en un cero para poder alojar la inicialización se cuenta con un equipo de 4G de RAM, es decir a 100000000.

# 3s-01.c

This program uses threads in order to divide the task of counting 3s in a given vector amongst different "working units", a.k.a. *threads*. Read and understand what this program does.

## Assignment

Include the code sentences used to estimate how much time takes to each thread to count the number of 3s in a given array.

Answer the following questions

- How many elements the large vector has?

Como se aclaró en una nota anterior, el equipo usado para las pruebas tiene 4G de RAM, esto me permite trabajar con una longitud de vector 100000000 (VECTOR_SIZE). Es decir, que a cada hilo se le asigna el procesamiento de un vector con una longitud de 12'500'000, como lo confirma la ejecución del programa.

- The program is correct? What is wrong with it? What value the program gets and what is the expected value?

El programa no es correcto, dado que no se obtiene el resultado esperado en el conteo del número de elementos del vector iguales al número 3. Se obtiene un valor muy por debajo del real en este conteo, esto se debe a que no se está manejando el “join” de lo hilos (`pthread_join`), lo que hace que no todos los hilos terminen su ejecución. En la mayoría de ocasiones solo se ha podido ver terminado el primer hilo, aunque otras pocas veces termina otro de los hilos.

# 3s-02.c

This version of the program is more accurate but it is not completely correct.

## Assignment

- Include the code sentences used to estimate how much time takes to each thread to count the number of 3s in a given array.
- What is wrong with this code this time?

Dado que se crean múltiples hilos que están accediendo al tiempo a la variable count, realizando una operación de incremento que no es atómica . Es decir que es un problema de concurrencia.

# 3s-03.c

This time the program is doing what we expect it has to do.

## Assignment

- Identify why this program is now doing right.

Por el uso del bloqueo de la variable, con ayuda del objeto mutex.

- Include the code sentences used to estimate how much time takes to each thread to count the number of 3s in a segment of the full array.
- How much time took to obtain the total count of 3s in the whole array.

El tiempo varía entre 312 y 223 ms. Que comparado con el que tomaba el programa secuencial es un poco mayor.

# 3s-04.c

The program has a little difference with the previous program.

## Assignment

- What is the difference between 3s-03.c and 3s-04.c?

En 3s-03.c se hacía un bloqueo para el recorrido completo del vector asignado a cada hilo. En cambio en 3s-04.c se bloquea solamente la operación de incrementar el contador.

- Compare elapsed time per thread during the counting process **and** the total time that all threads took for counting the number of 3s in the whole array. Run every program (3s-03 and 3s-04) three times and compute the average time per program. Present your results and explain them.

| Tiempo | 3s-03 (ms) | 3s-04 (ms) |
|---|---|---|
| 1 | 316.8 | 765.6 |
| 2 | 312.4 | 755.8 |
| 3 | 323.1 | 750.8 |
| promedio | 317.4 | 757.4 |

La tabla anterior me muestra que los tiempos de la solución presentada en el archivo 3s-04.c toma más tiempo que las que se presenta en 3s-03.c. Esto se explica porque en el 3s-04.c planteó una solución que implica muchos más bloqueos que en el otro enfoque; y dado que cada instrucción de bloqueo y desbloqueo utiliza ciclos de procesamiento, se incrementará el tiempo final del conteo.

# 3s-05.c

This time this solution is correct and exhibits a better performance when it is compared with the previous programs.

# Assignment

- Where the success of this program resides?

Ahora se reduce el bloqueo a una sola instrucción utilizando un contador privado para cada hilo. Es decir que el tiempo por el que está bloqueada la variable `count`, así como el número de veces, se reduce significativamente con respecto al ejemplo anterior.

- Compare this program execution and compare its performance with previous instances and write your observations.

Se reduce considerablemente, incluso se logra superar el tiempo que gastó el programa secuencial. Es un tiempo cercano a los 70 ms, que es alrededor de cinco veces menor que el que tomó el programa del ejemplo 3s-03.