



# Computación Paralela

Alumna:

Esther Milagros Bautista Peralata

Semestre: *VIII - semestre*

Docente:	<i>Fred Torres Cruz</i>
Facultad:	Ingeniería Estadística e Informática
Año académico:	October 26, 2022- II

---

## Contents

<b>1</b>	<b>Ejercicio n°1</b>	<b>1</b>
<b>2</b>	<b>Ejercicio n°2</b>	<b>1</b>
<b>3</b>	<b>Ejercicio n°3</b>	<b>5</b>
<b>4</b>	<b>Evidencia del trabajo en LATEX</b>	<b>10</b>

## 1 Ejercicio n°1

El proceso de cifrado de un sistema de contraseñas está subdividido en dos (02) partes x1(SHA-256),x2(MD5) iterativamente y secuencialmente, se asume que cada una de las partes es independiente, además también se conoce que x2 consume el 35% del tiempo total de computación. Si tenemos 02 opciones de mejora:

F1 = Optimización de x1 : 8 veces más rápido.

F2 = Optimización de x2 : 3 veces más rápido.

¿Cuál es la más óptima y por qué?

### Solución

$$X = X_1 + X_2$$

$$X = 65\% + 35\%$$

$$F_1 = \frac{65\%}{8} + 35\% = 43.1\%$$

$$F_2 = 65\% + \frac{35\%}{3} = 76.7\%$$

Es mejor optimizar F1 por que reduce el porcentaje del consumo del tiempo.

## 2 Ejercicio n°2

Se requiere una mejora en la velocidad de un CPU mediante un procedimiento de overclocking , pero esta mejora solo impactaría en procesadores con una velocidad mayor a 3.2 GHz y en un 20 % ¿Calcule la aceleración para los siguientes casos?

a) Procesador A : 2.86 GHz

Se queda con la misma velocidad 2.86 GHz

b) Procesador A : 3.58 GHz

$$3.58 + \left( \frac{20}{100} \times 3.58 \right) = 4.296GHz$$

c) Procesador A : 3.14 GHz

Se queda con la misma velocidad 3.14 GHz

d) Procesador A : 3.81 GHz

$$3.81 + \left( \frac{20}{100} \times 3.81 \right) = 4.572GHz$$

## CÓDIGO EN SERIE USANDO UNA FUNCIÓN

```
import time

def proceso():
    time_ini = time.time()
    print("---- Inicio ----")
    time.sleep(1)

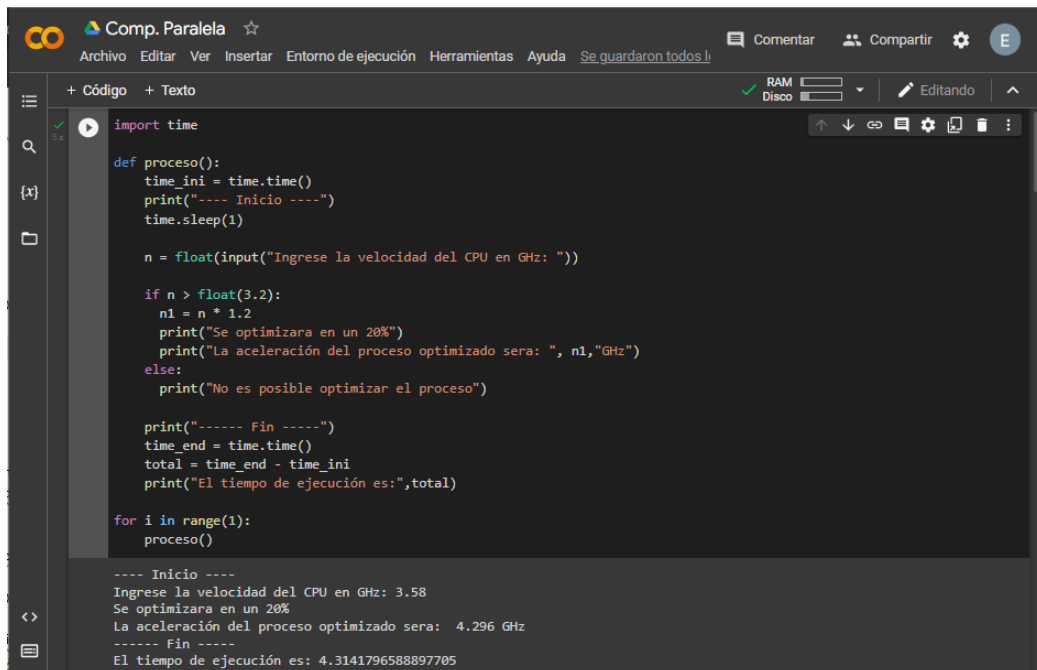
    n = float(input("Ingrese la velocidad del CPU en GHz: "))

    if n > float(3.2):
        n1 = n * 1.2
        print("Se optimizara en un 20%")
        print("La aceleración del proceso optimizado sera: ", n1,"GHz")
    else:
        print("No es posible optimizar el proceso")

    print("----- Fin -----")
    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:",total)

for i in range(1):
    proceso()
```

### Salida del código en serie usando una función



```
Comp. Paralela
Archivo  Editar  Ver  Insertar  Entorno de ejecución  Herramientas  Ayuda  Se guardaron todos l...
+ Código  + Texto
import time

def proceso():
    time_ini = time.time()
    print("---- Inicio ----")
    time.sleep(1)

    n = float(input("Ingrese la velocidad del CPU en GHz: "))

    if n > float(3.2):
        n1 = n * 1.2
        print("Se optimizara en un 20%")
        print("La aceleración del proceso optimizado sera: ", n1,"GHz")
    else:
        print("No es posible optimizar el proceso")

    print("----- Fin -----")
    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:",total)

for i in range(1):
    proceso()

---- Inicio ----
Ingrese la velocidad del CPU en GHz: 3.58
Se optimizara en un 20%
La aceleración del proceso optimizado sera: 4.296 GHz
----- Fin -----
El tiempo de ejecución es: 4.314179658897705
```

Figure 1: Salida del código

## **CÓDIGO USANDO HILOS (THREAD)**

```
from threading import Thread
import time

def proceso():
    time_ini = time.time()
    print("---- Inicio ----")
    time.sleep(1)

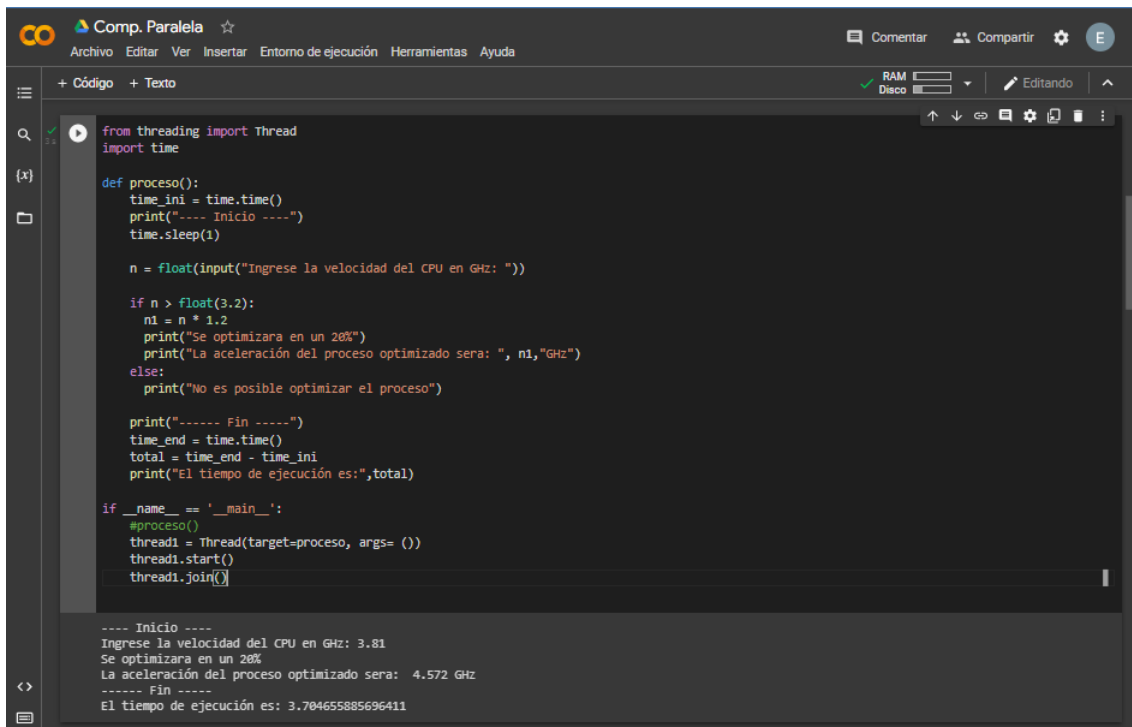
    n = float(input("Ingrese la velocidad del CPU en GHz: "))

    if n > float(3.2):
        n1 = n * 1.2
        print("Se optimizara en un 20%")
        print("La aceleración del proceso optimizado sera: ", n1, "GHz")
    else:
        print("No es posible optimizar el proceso")

    print("----- Fin -----")
    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:", total)

if __name__ == '__main__':
    #proceso()
    thread1 = Thread(target=proceso, args= ())
    thread1.start()
    thread1.join()
```

**Salida del código usando hilos (thread)**



The screenshot shows the 'Comp. Paralela' IDE interface. The top menu bar includes 'Archivo', 'Editar', 'Ver', 'Insertar', 'Entorno de ejecución', 'Herramientas', and 'Ayuda'. The toolbar has icons for 'Comentar', 'Compartir', and 'E'. The main editor displays Python code for a threaded process. The code imports 'Thread' and 'time' from the 'threading' module. It defines a 'proceso()' function that prints 'Inicio', sleeps for 1 second, prompts for CPU speed, and checks if it's greater than 3.2 GHz. If so, it calculates a 20% increase and prints the optimized speed. It then prints 'Fin' and the total execution time. The main block uses 'Thread' to run 'proceso()' and joins it. The output at the bottom shows the execution results: 'Inicio', 'Ingrese la velocidad del CPU en GHz: 3.81', 'Se optimizara en un 20%', 'La aceleración del proceso optimizado sera: 4.572 GHz', 'Fin', and 'El tiempo de ejecución es: 3.704655885696411'.

```
from threading import Thread
import time

def proceso():
    time_ini = time.time()
    print("---- Inicio ----")
    time.sleep(1)

    n = float(input("Ingrese la velocidad del CPU en GHz: "))

    if n > float(3.2):
        n1 = n * 1.2
        print("Se optimizara en un 20%")
        print("La aceleración del proceso optimizado sera: ", n1,"GHz")
    else:
        print("No es posible optimizar el proceso")

    print("----- Fin -----")
    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:",total)

if __name__ == '__main__':
    #proceso()
    thread1 = Thread(target=proceso, args= ())
    thread1.start()
    thread1.join()
```

---- Inicio ----  
Ingrese la velocidad del CPU en GHz: 3.81  
Se optimizara en un 20%  
La aceleración del proceso optimizado sera: 4.572 GHz  
----- Fin -----  
El tiempo de ejecución es: 3.704655885696411

Figure 2: Salida del código

## CÓDIGO USANDO PROCESOS

```
import time #pip install time
import multiprocessing
from multiprocessing import Process

def mundo(n):
    time_ini = time.time()
    print("---- Inicio ----")
    time.sleep(1)

    if n > float(3.2):
        n1 = n * float(1.2)
        print("Se optimizara en 20%")
        print("La aceleración del proceso optimizado sera: ", n1,"GHz")
    else:
        print("No es posible optimizar el proceso")

    print("----- Fin -----")
    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:",total)

if __name__ == '__main__':
    n = float(input("Ingrese la velocidad del CPU en GHz: "))
```

```

multiprocessing.set_start_method('spawn')
p = Process(target=mundo, args=(n,)) #Proceso
p.start()

```

## Salida del código usando procesos

The screenshot shows the Visual Studio Code interface with a Python file named 'procesos.py' open. The code defines a function 'mundo(n)' that simulates a process optimization. The terminal output shows the execution of the script, which prompts for CPU velocity and displays the results of the optimization process.

```

D:\> procesos.py > mundo
1 import time #pip install time
2 import multiprocessing
3 from multiprocessing import Process
4
5 def mundo(n):
6     time_ini = time.time()
7     print("---- Inicio ----")
8     time.sleep(1)
9
10    if n > float(3.2):
11        n1 = n * float(1.2)
12        print("Se optimizara en 20%")
13        print("La aceleración del proceso optimizado sera: ", n1,"GHz")
14    else:
15        print("No es posible optimizar el proceso")
16
17    print("----- Fin -----")

```

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\MILAGROS> & C:/Users/MILAGROS/AppData/Local/Programs/Python/Python310/python.exe d:/procesos.py
Ingresa la velocidad del CPU en GHz: 3.81
---- Inicio ----
Se optimizara en 20%
La aceleración del proceso optimizado sera: 4.572 GHz
----- Fin -----
El tiempo de ejecución es: 1.006561279296875
PS C:\Users\MILAGROS>

```

Figure 3: Salida del código

## 3 Ejercicio n°3

Realizar una función que calcule la distancia entre dos puntos en una implementación lineal y en una implementación utilizando hilos, así mismo evaluar la diferencia entre estas.

$$p = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

### Solución

#### CÓDIGO EN UNA IMPLEMENTACIÓN LINEAL

```

from math import sqrt
import time

```

```

def distancial():

```

```

time_ini = time.time()
print("---- Inicio ----")
time.sleep(1)

print("Ingrese los valores de punto 1")
x1 = float(input("Ingrese el valor de x1: "))
y1 = float(input("Ingrese el valor de y1: "))
print("Ingrese los valores de punto 2")
x2 = float(input("Ingrese el valor de x2: "))
y2 = float(input("Ingrese el valor de y2: "))

distancia = sqrt((x2-x1)**2 + (y2-y1)**2)
print("La distancia entre los 2 puntos es: ", distancia)

print("----- Fin -----")
time_end = time.time()
total = time_end - time_ini
print("El tiempo de ejecución es:",total)

for i in range(1):
    distancial()

```

### Salida del código en una implementación lineal

```

from math import sqrt
import time

def distancia1():
    time_ini = time.time()
    print("---- Inicio ----")
    time.sleep(1)

    print("Ingrese los valores de punto 1")
    x1 = float(input("Ingrese el valor de x1: "))
    y1 = float(input("Ingrese el valor de y1: "))
    print("Ingrese los valores de punto 2")
    x2 = float(input("Ingrese el valor de x2: "))
    y2 = float(input("Ingrese el valor de y2: "))

    distancia = sqrt((x2-x1)**2 + (y2-y1)**2)
    print("La distancia entre los 2 puntos es: ", distancia)

    print("----- Fin -----")
    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:",total)

for i in range(1):
    distancia1()

```

```

---- Inicio ----
Ingrese los valores de punto 1
Ingrese el valor de x1: -4
Ingrese el valor de y1: -3
Ingrese los valores de punto 2
Ingrese el valor de x2: 2
Ingrese el valor de y2: 5
La distancia entre los 2 puntos es: 10.0
----- Fin -----
El tiempo de ejecución es: 12.877672910690308

```

Figure 4: Salida del código



## CÓDIGO IMPLEMENTANDO HILOS (THREAD)

```
from threading import Thread
import time

def distancia1():
    time_ini = time.time()

    print("---- Inicio ----")
    time.sleep(1)

    print("Ingrese los valores de punto 1")
    x1 = float(input("Ingrese el valor de x1: "))
    y1 = float(input("Ingrese el valor de y1: "))
    print("Ingrese los valores de punto 2")
    x2 = float(input("Ingrese el valor de x2: "))
    y2 = float(input("Ingrese el valor de y2: "))

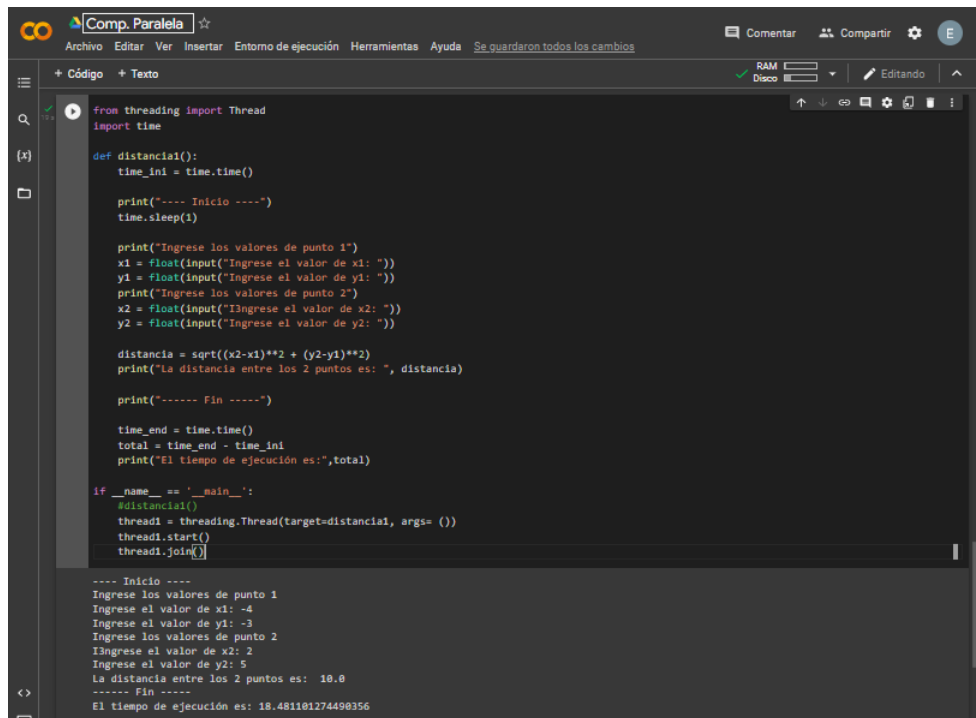
    distancia = sqrt((x2-x1)**2 + (y2-y1)**2)
    print("La distancia entre los 2 puntos es: ", distancia)

    print("----- Fin -----")

    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:", total)

if __name__ == '__main__':
    #distancia1()
    thread1 = threading.Thread(target=distancia1, args= ())
    thread1.start()
    thread1.join()
```

**Salida del código usando hilos (thread)**



```
from threading import Thread
import time

def distancia1():
    time_ini = time.time()

    print("---- Inicio ----")
    time.sleep(1)

    print("Ingrese los valores de punto 1")
    x1 = float(input("Ingrese el valor de x1: "))
    y1 = float(input("Ingrese el valor de y1: "))
    print("Ingrese los valores de punto 2")
    x2 = float(input("Ingrese el valor de x2: "))
    y2 = float(input("Ingrese el valor de y2: "))

    distancia = sqrt((x2-x1)**2 + (y2-y1)**2)
    print("La distancia entre los 2 puntos es: ", distancia)

    print("----- Fin -----")

    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:",total)

if __name__ == '__main__':
    distancia1()
    thread1 = threading.Thread(target=distancia1, args= ())
    thread1.start()
    thread1.join()
```

---- Inicio ----  
Ingrese los valores de punto 1  
Ingrese el valor de x1: -4  
Ingrese el valor de y1: -3  
Ingrese los valores de punto 2  
Ingrese el valor de x2: 2  
Ingrese el valor de y2: 5  
La distancia entre los 2 puntos es: 10.0  
----- Fin -----  
El tiempo de ejecución es: 18.481181274490356

Figure 5: Salida del código

## CÓDIGO USANDO PROCESOS

```
import time
import multiprocessing
from multiprocessing import Process
from math import sqrt

def mundo(x1,y1,x2,y2):
    time_ini = time.time()
    print("---- Inicio ----")
    time.sleep(1)

    distancia = sqrt((x2-x1)**2 + (y2-y1)**2)
    print("La distancia entre los 2 puntos es: ", distancia)

    print("----- Fin -----")
    time_end = time.time()
    total = time_end - time_ini
    print("El tiempo de ejecución es:",total)

if __name__ == '__main__':
    print("Ingrese los valores de punto 1")
    x1 = float(input("Ingrese el valor de x1: "))
    y1 = float(input("Ingrese el valor de y1: "))
    print("Ingrese los valores de punto 2")
```

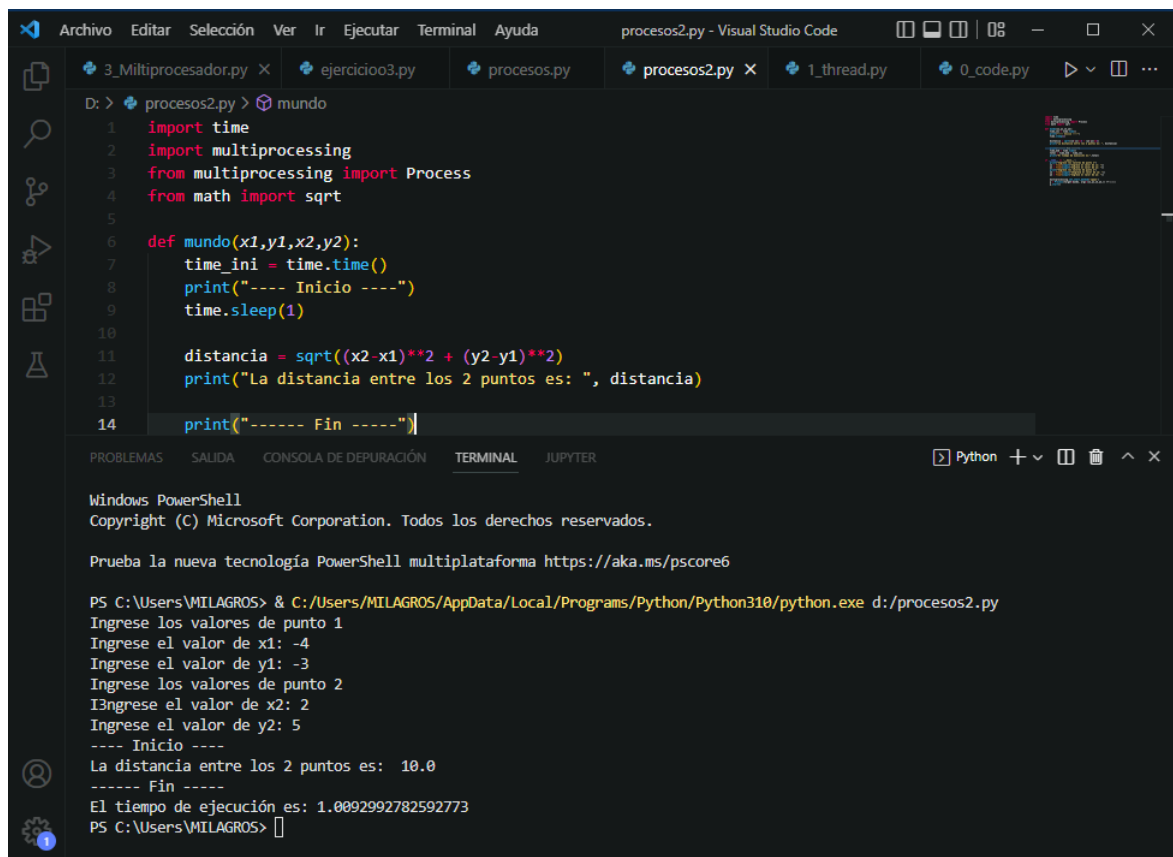
```

x2 = float(input("Ingrese el valor de x2: "))
y2 = float(input("Ingrese el valor de y2: "))

multiprocessing.set_start_method('spawn')
p = Process(target=mundo, args=(x1,y1,x2,y2,)) #Proceso
p.start()

```

## Salida del código usando procesos



The screenshot shows the Visual Studio Code interface with a Python file named 'procesos2.py' open. The code defines a function 'mundo' that calculates the distance between two points (x1, y1) and (x2, y2) using the Pythagorean theorem. It also includes a sleep function to simulate a delay. The terminal output shows the execution of the script, where the user enters values for x1, y1, x2, and y2, and the program outputs the calculated distance and the execution time.

```

D:\> cd D:\procesos2.py & python mundo
1 import time
2 import multiprocessing
3 from multiprocessing import Process
4 from math import sqrt
5
6 def mundo(x1,y1,x2,y2):
7     time_ini = time.time()
8     print("---- Inicio ----")
9     time.sleep(1)
10
11     distancia = sqrt((x2-x1)**2 + (y2-y1)**2)
12     print("La distancia entre los 2 puntos es: ", distancia)
13
14     print("----- Fin -----")

```

```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\MILAGROS> & C:\Users\MILAGROS\AppData\Local\Programs\Python\Python310\python.exe d:/procesos2.py
Ingrese los valores de punto 1
Ingrese el valor de x1: -4
Ingrese el valor de y1: -3
Ingrese los valores de punto 2
Ingrese el valor de x2: 2
Ingrese el valor de y2: 5
---- Inicio ----
La distancia entre los 2 puntos es: 10.0
----- Fin -----
El tiempo de ejecución es: 1.0092992782592773
PS C:\Users\MILAGROS>

```

**Figure 6:** Salida del código

En conclusión, un proceso no es más que la instancia de un programa, el cual es creado y controlado por el sistema operativo. Los procesos son entidades independientes y no pueden compartir información entre ellos. Para que un proceso pueda ser ejecutado es necesario que este posea por lo menos un thread.

Por otro lado, un thread es la unidad más pequeña a la cual un procesador puede asignar tiempo. Los threads poseerán la secuencia más pequeña de instrucciones a ejecutar. Los threads se crean, ejecutan y mueren dentro de los procesos, siendo capaces de compartir información entre ellos.

Con los threads y los procesos seremos capaces de implementar la programación concurrente, y, dependiendo de la cantidad de procesadores la programación en paralelo.

## 4 Evidencia del trabajo en LATEX

The screenshot shows the Overleaf LaTeX editor interface. The top bar includes navigation and utility icons. The left sidebar shows the file outline with 'Ejercicio nº1' selected. The main editor displays the LaTeX source code for 'Ejercicio nº1', which describes a password cracking process and asks for optimization. The right pane shows the compiled PDF output, which includes the problem statement, a solution section with mathematical calculations for F1 and F2, and a second exercise 'Ejercicio nº2'.

**1 Ejercicio nº1**

El proceso de cifrado de un sistema de contraseñas está subdividido en dos (02) partes x1(SHA-256),x2(MD5) iterativamente y secuencialmente, se asume que cada una de las partes es independiente, además también se conoce que x2 consume el 35% del tiempo total de computación. Si tenemos 02 opciones de mejora:

F1 = Optimización de x1 : 8 veces más rápido.  
F2 = Optimización de x2 : 3 veces más rápido.

¿Cuál es a más óptima y por qué?

**Solución**

$$X = X_1 + X_2$$
$$X = 65\% + 35\%$$
$$F_1 = \frac{65\%}{8} + 35\% = 43.1\%$$
$$F_2 = 65\% + \frac{35\%}{3} = 76.7\%$$

Es mejor optimizar F1 por que reduce el porcentaje del consumo del tiempo.

**2 Ejercicio nº2**

Se requiere una mejora en la velocidad de un CPU mediante un procedimiento de overclocking , pero esta mejora solo impactaría en procesadores con una velocidad mayor a 3.2 GHz y en un 20 % ¿Calcule la aceleración para los siguientes casos?

a) Procesador A : 2.86 GHz

Se queda con la misma velocidad 2.86 GHz

b) Procesador A : 3.58 GHz

$$3.58 + \left(\frac{20}{100} \times 4.296\right) = 4.296 \text{ GHz}$$

Figure 7: <https://es.overleaf.com/read/qyhsxzmxcdyz>