Experimento 4

GÓMEZ EDGAR, Universidad Central de Venezuela, Venezuela

Este experimento implementa un sistema de procesos concurrentes con comunicación padre-hijo en Python, enfocándose en la visualización y monitoreo de procesos a través del Administrador de Tareas del sistema operativo. Se extiende el tiempo de ejecución de los procesos hijos a 120 segundos para facilitar la observación de su comportamiento en tiempo real. Se demuestra la correlación entre los PID reportados por la aplicación y los mostrados en el Administrador de Tareas, validando la creación y gestión efectiva de procesos en sistemas distribuidos.

CCS Concepts: • **Organización de sistemas informáticos → Sistemas Distribuidos**; *Hilos en software*; *Procesos en software*.

Additional Key Words and Phrases: Sistemas Distribuidos, Arquitectura de Software, procesos en software, hilos en software

ACM Reference Format:
GÓMEZ EDGAR. 2025. Experimento 4. *Proc. ACM Hum.-Comput. Interact.* 1, 1, Article 1 (December 2025), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1. Introducción

Este experimento extiende los conceptos previos de comunicación entre procesos incorporando una dimensión de observación del sistema operativo. Mientras que experimentos anteriores demostraron la creación y comunicación de procesos, este trabajo se enfoca en correlacionar la información interna de la aplicación (PID, estados) con la representación externa en el Administrador de Tareas. Se implementa un tiempo de ejecución prolongado (120 segundos) en los procesos hijos para permitir una observación detallada de su ciclo de vida.

2. Proceso Padre

El código del padre se mantiene igual de nuestro experimentos anteriores. El proceso padre crea multiples hijos y recibe información de los hijos.

```
import os
import subprocess

print(f"Hola, soy el proceso padre. Mi PID es: {os.getpid()}")
print("Creando procesos hijos...")

array = []
for i in range(5):
# Iniciar el proceso hijo con stdout por pipe (el hijo enviará datos al padre)
nuevo = subprocess.Popen(
["python", "hijo.py"],
stdout=subprocess.PIPE,  # Redirigir stdout del hijo a una tubería
stderr=subprocess.PIPE,
text=True,  # Trabajar con texto en lugar de bytes
bufsize=1,  # Buffer en línea (line-buffered)
universal_newlines=True  # Compatibilidad con nuevas líneas
)
```

Author's Contact Information: GÓMEZ EDGAR, Universidad Central de Venezuela, Caracas, Venezuela.

2025. ACM 2573-0142/2025/12-ART1
https://doi.org/10.1145/nnnnnnn.nnnnnnn

```
array.append(nuevo)

for p in array:
print(f"El proceso hijo se ha creado con el PID: {p.pid}")

# Esperar a cada hijo y leer lo que envió por stdout
for p in array:
p.wait()  # espera a que el hijo termine

if p.stdout:
salida = p.stdout.read()  # leer todo lo enviado por el hijo
p.stdout.close()
else:
salida = ""

print(f"Salida del hijo PID {p.pid}:\n{salida.strip()}")
print(f"El proceso hijo con PID {p.pid} ha terminado.")

print("Adiós, soy el proceso padre.")
```

3. Proceso Hijo

Para poder visualizar los porcesos en el Administrador de Tareas aumentamos el tiempo de espera que simula una carga de trabajo en el proceso hijo a dos minutos.

```
import os
import time
import sys

sys.stdout.write(f"Hola, soy el proceso hijo. Mi PID es: {os.getpid()}.\n")
sys.stdout.flush()
time.sleep(120) # Simula una tarea que toma 120 segundos
sys.stdout.write("El proceso hijo ha terminado su tarea.\n")
sys.stdout.flush()
```

4. Resultados de la ejecución

Ejecutando el código, podemos observar en el adminstrador de tareas, Fig 1, los PID de todos los procesos los del padre conjuntamente con los hijos, podemos corroborar los PID con la salida de la consola Fig 2, donde podemos observar que los PID coinciden.

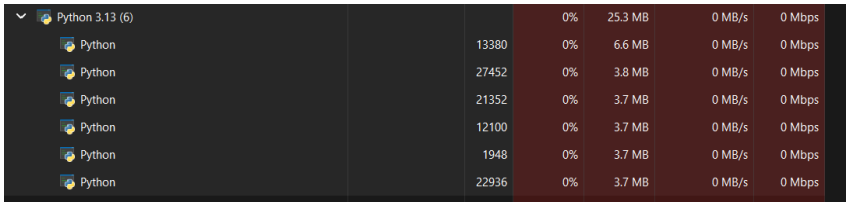

Fig. 1. PID de los procesos en el administrador de tareas.

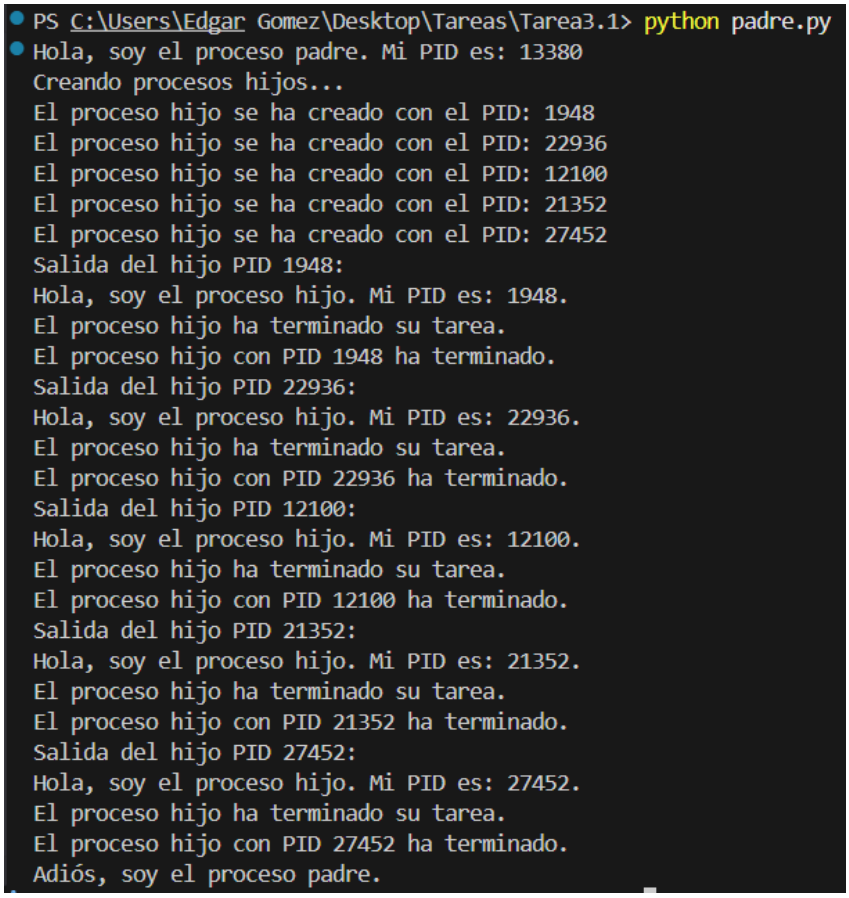

Fig. 2. Pid de los procesos en la salida del terminal.

5. Conclusión

El experimento demostró exitosamente la correlación entre los procesos gestionados por la aplicación Python y su representación en el Administrador de Tareas del sistema operativo. Se verificó que los PID reportados por la aplicación coinciden exactamente con los mostrados en las herramientas del sistema, validando la creación y

gestión adecuada de los procesos. La extensión del tiempo de ejecución a 120 segundos permitió una observación clara del ciclo de vida de los procesos, confirmando que permanecen activos y gestionables durante su ejecución.