Informe de Laboratorio 1: Gestión de Procesos

1. Introducción

En este laboratorio se analizan los estados de los procesos dentro de un sistema operativo, la planificación de la CPU y la simulación de condiciones de deadlock. Se utilizaron scripts en Python para observar y documentar el comportamiento del sistema bajo diferentes escenarios de uso de procesos, dentro de una máquina virtual configurada en VirtualBox con Windows.

2. Estados de Proceso

Se ejecutó un script de Python que simula las transiciones de un proceso por los estados: Nuevo, Listo, Ejecutando, Bloqueado y Terminado. Se midieron los tiempos de transición entre cada uno usando la librería time.

📊 Tiempos de ejecución registrados:

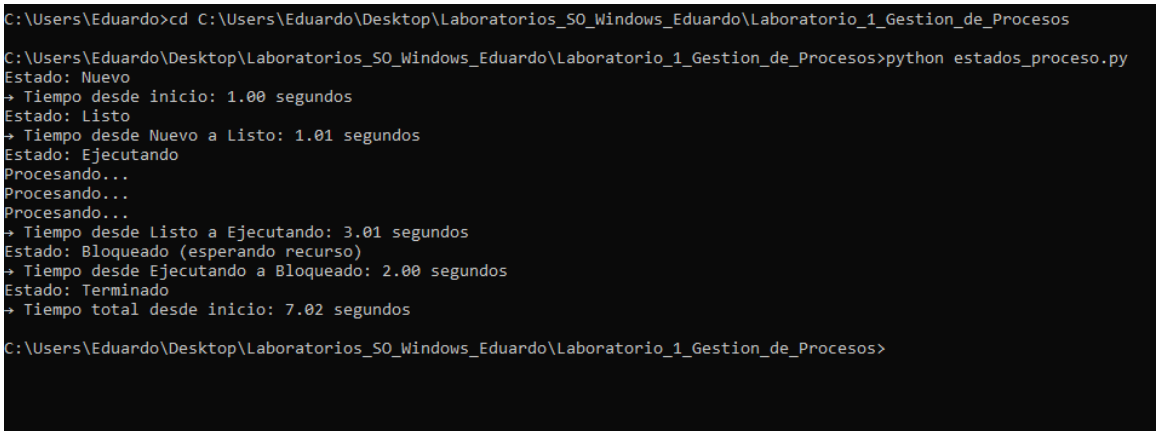
- Inicio → Espera: 01:00 segundos

- Espera → Listo: 01.01 segundos

- Listo → Ejecutando : 03.01 segundos

- Ejecutando → bloqueado: 02.00 segundos

- Tiempo total del proceso: 07.02 segundos



*estados_proceso: Bloc de notas

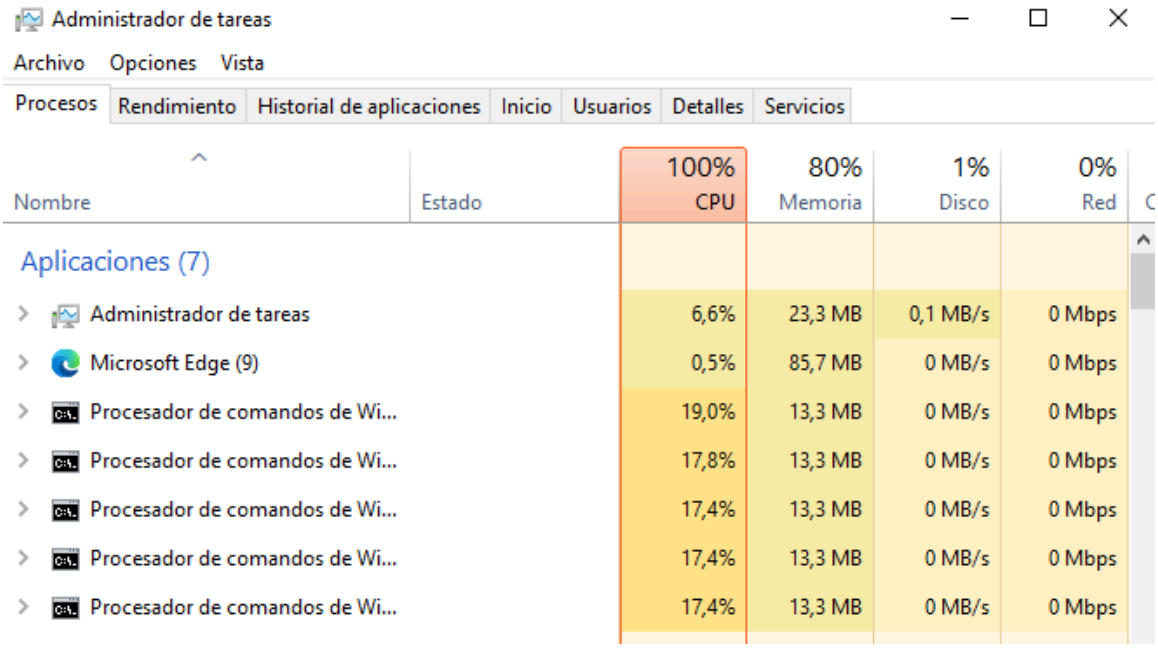
Archivo Edición Formato Ver Ayuda

```
import time

start = time.time()
print("Estado: Nuevo")
time.sleep(1)

t_nuevo = time.time()
print(f"→ Tiempo desde inicio: {t_nuevo - start:.2f} segundos")

print("Estado: Listo")
time.sleep(1)

t_listo = time.time()
print(f"→ Tiempo desde Nuevo a Listo: {t_listo - t_nuevo:.2f} segundos")

print("Estado: Ejecutando")
for i in range(3):
    print("Procesando...")
    time.sleep(1)

t_ejecutando = time.time()
print(f"→ Tiempo desde Listo a Ejecutando: {t_ejecutando - t_listo:.2f} segundos")

print("Estado: Bloqueado (esperando recurso)")
time.sleep(2)

t_bloqueado = time.time()
print(f"→ Tiempo desde Ejecutando a Bloqueado: {t_bloqueado - t_ejecutando:.2f} segundos")

print("Estado: Terminado")
t_terminado = time.time()
print(f"→ Tiempo total desde inicio: {t_terminado - start:.2f} segundos")
```

Línea 33, columna 1 | 100% | Windows (CRLF) | UTF-8

3. Planificación de CPU (Scheduling)

Se ejecutaron 5 procesos intensivos simultáneamente en la máquina virtual. El uso de CPU fue observado mediante el Administrador de tareas. El sistema operativo distribuyó el tiempo de CPU entre ellos, de forma similar a una planificación Round Robin, donde cada proceso obtuvo una parte equitativa del CPU.



*Sin título: Bloc de notas

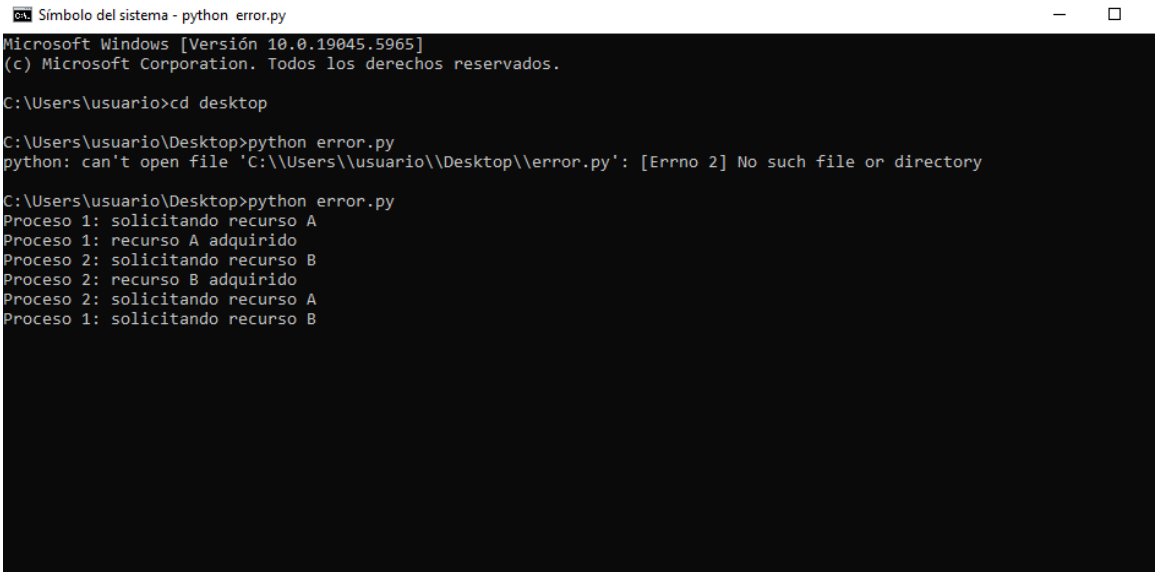
Archivo Edición Formato Ver Ayuda

```
import math

print("Iniciando proceso pesado...")
while True:
    [math.sqrt(i) for i in range(1_000_000)]
```

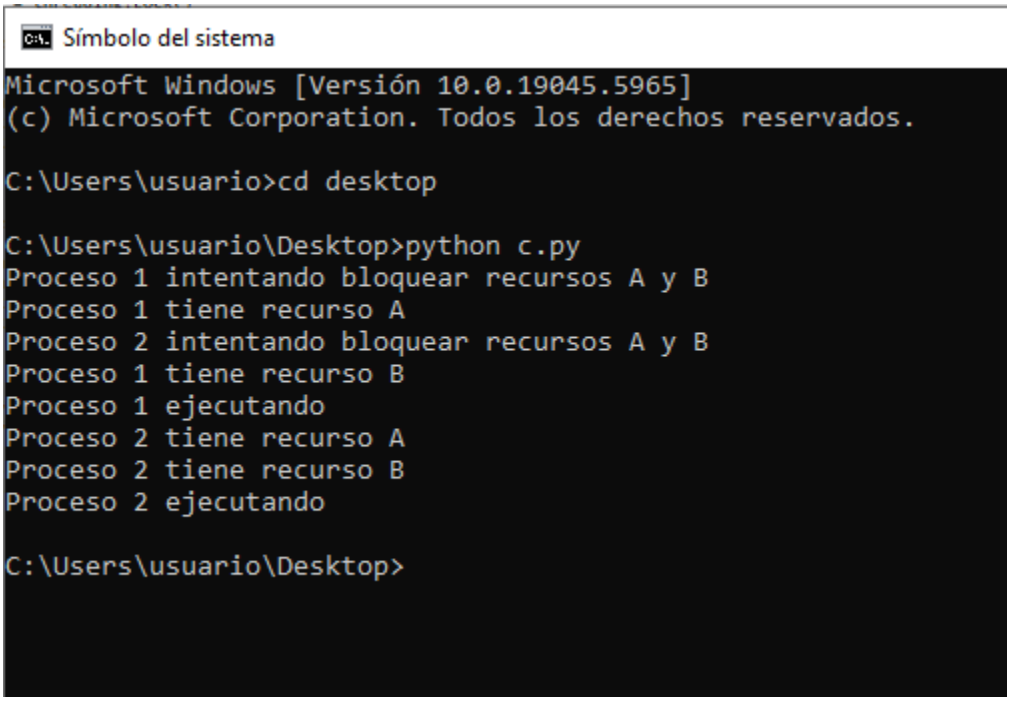
4. Simulación de Deadlock

Se utilizó un script de Python con hilos y bloqueos mutuos para provocar una condición de interbloqueo (deadlock). Ambos hilos quedaron esperando indefinidamente, demostrando cómo el sistema puede verse afectado si no se maneja correctamente el acceso a recursos compartidos.

```
import threading
import time

# Recursos simulados
recurso_A = threading.Lock()
recurso_B = threading.Lock()

def proceso1():
    print("Proceso 1: solicitando recurso A")
    recurso_A.acquire()
    print("Proceso 1: recurso A adquirido")
    time.sleep(1)

    print("Proceso 1: solicitando recurso B")
    recurso_B.acquire()
    print("Proceso 1: recurso B adquirido")

    print("Proceso 1: ejecutando con ambos recursos")
    recurso_B.release()
    recurso_A.release()

def proceso2():
    print("Proceso 2: solicitando recurso B")
    recurso_B.acquire()
    print("Proceso 2: recurso B adquirido")
    time.sleep(1)

    print("Proceso 2: solicitando recurso A")
    recurso_A.acquire()
    print("Proceso 2: recurso A adquirido")

    print("Proceso 2: ejecutando con ambos recursos")
    recurso_A.release()
    recurso_B.release()

# Crear hilos
t1 = threading.Thread(target=proceso1)
t2 = threading.Thread(target=proceso2)

# Iniciar ambos hilos
t1.start()
t2.start()
```



"La solución a este problema es que ambos procesos accedan en el mismo orden primero A y luego B por lo que no hay riesgo de bloqueo mutuo"



5. Conclusión

Este laboratorio permitió entender cómo funcionan los procesos en un sistema operativo real, cómo se distribuyen los recursos del CPU y qué consecuencias puede tener una mala

sincronización entre procesos. La observación desde una máquina virtual permitió replicar condiciones reales de forma controlada y segura.