UNIVERSIDAD DE LA INTEGRACIÓN DE LAS AMÉRICAS

FACULTAD DE INGENIERÍA.

MATERIA

Sistemas Operativos.

TÍTULO

Laboratorio de Análisis de Sistemas Operativos.

INFORME DE LABORATORIO N°1:

Gestión de Procesos

Mg. Alan Vladimir Dioses Echegaray.

Lucio Vera.

ESTUDIANTE:

Jannely Magalí Guillén Capdevila.

Asunción- Paraguay.

2025.

Introducción.

El presente informe detalla los experimentos realizados en el marco del primer laboratorio sobre gestión de procesos en un sistema operativo Windows. La gestión de procesos es una de las funciones más críticas de cualquier sistema operativo moderno, ya que es responsable de administrar la ejecución de programas, asignar recursos y garantizar que el sistema se mantenga estable y responsivo. Este laboratorio busca observar de manera práctica los conceptos teóricos de estados de procesos, la planificación del CPU y los conflictos de recursos como el interbloqueo (deadlock).

Materiales y Métodos.

Para la realización de este laboratorio, se utilizó un entorno controlado dentro de una máquina virtual para no comprometer el sistema anfitrión, de acuerdo con los requerimientos del proyecto. La configuración utilizada fue:

Software de Virtualización: Oracle VM VirtualBox.

Sistema Operativo Anfitrión: Windows 11 Home.

Sistema Operativo Invitado (VM): Windows 10 Pro.

Herramientas de Análisis: Administrador de Tareas de Windows.

Scripts de Prueba: Se desarrollaron scripts personalizados en Python para simular cargas de trabajo y estados de procesos específicos.

Desarrollo y Resultados.

A continuación, se describen los procedimientos y los resultados obtenidos en cada una de las tres fases del laboratorio.

Estados de Procesos.

Se creó un script de Python (proceso_estados.py) diseñado para transitar explícitamente por los diferentes estados de un proceso.

Estado Nuevo/Listo: Se preparó el comando de ejecución en la terminal. En este punto, el sistema operativo ha creado la estructura del proceso, pero espera la señal para moverlo a la cola de "Listos" y posteriormente asignarle la CPU.

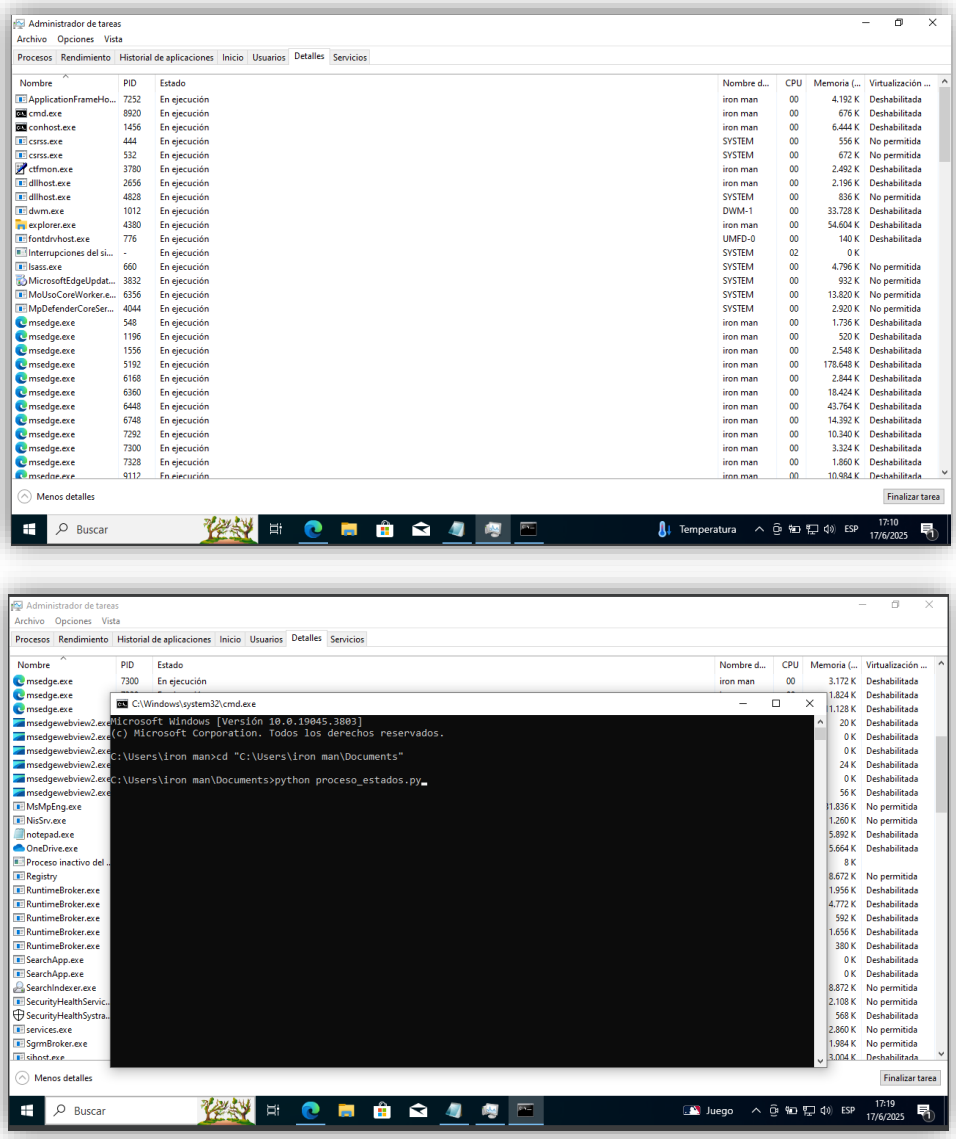

Figura 1. Preparación del proceso antes de su ejecución.

Estado Ejecutando: Al ejecutar el script, este realizó un bucle de cálculo intensivo. El Administrador de Tareas mostró un consumo de CPU significativo para el proceso python.exe, confirmando su estado de "Ejecutando".

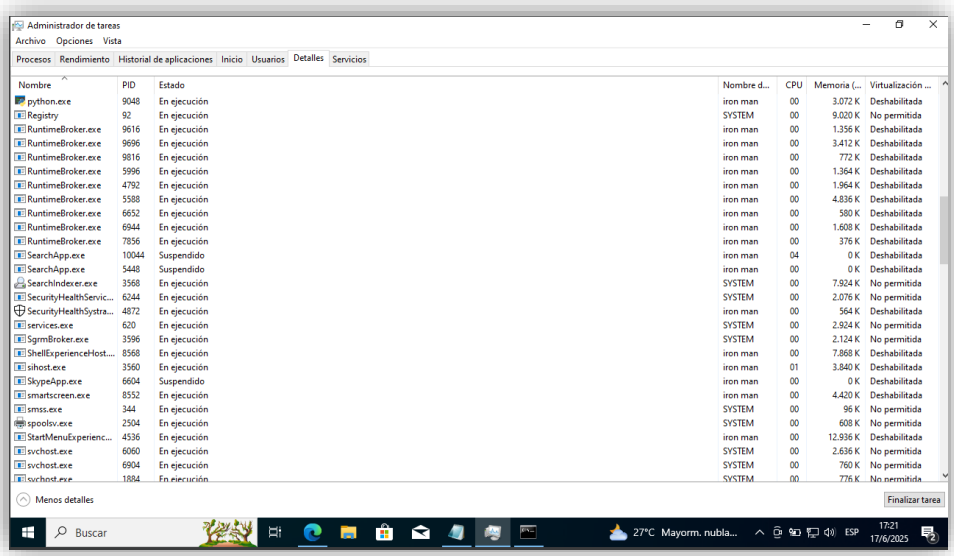

Figura 2. Proceso python.exe en estado de ejecución.

Estado Bloqueado: El script se detuvo a la espera de una entrada por teclado (input()). El consumo de CPU del proceso descendió a 0%, ya que estaba bloqueado esperando un evento externo (la pulsación de la tecla Enter).

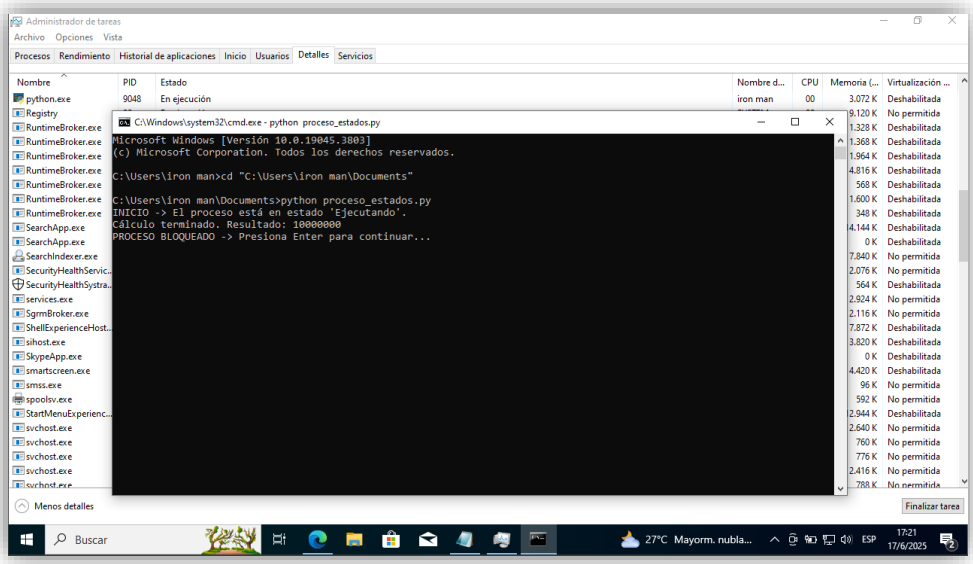

Figura 3. Proceso python.exe en estado bloqueado.

Estado Terminado: Una vez que el script completó su ejecución, el proceso python.exe desapareció de la lista del Administrador de Tareas, indicando que sus recursos fueron liberados y el proceso finalizó.

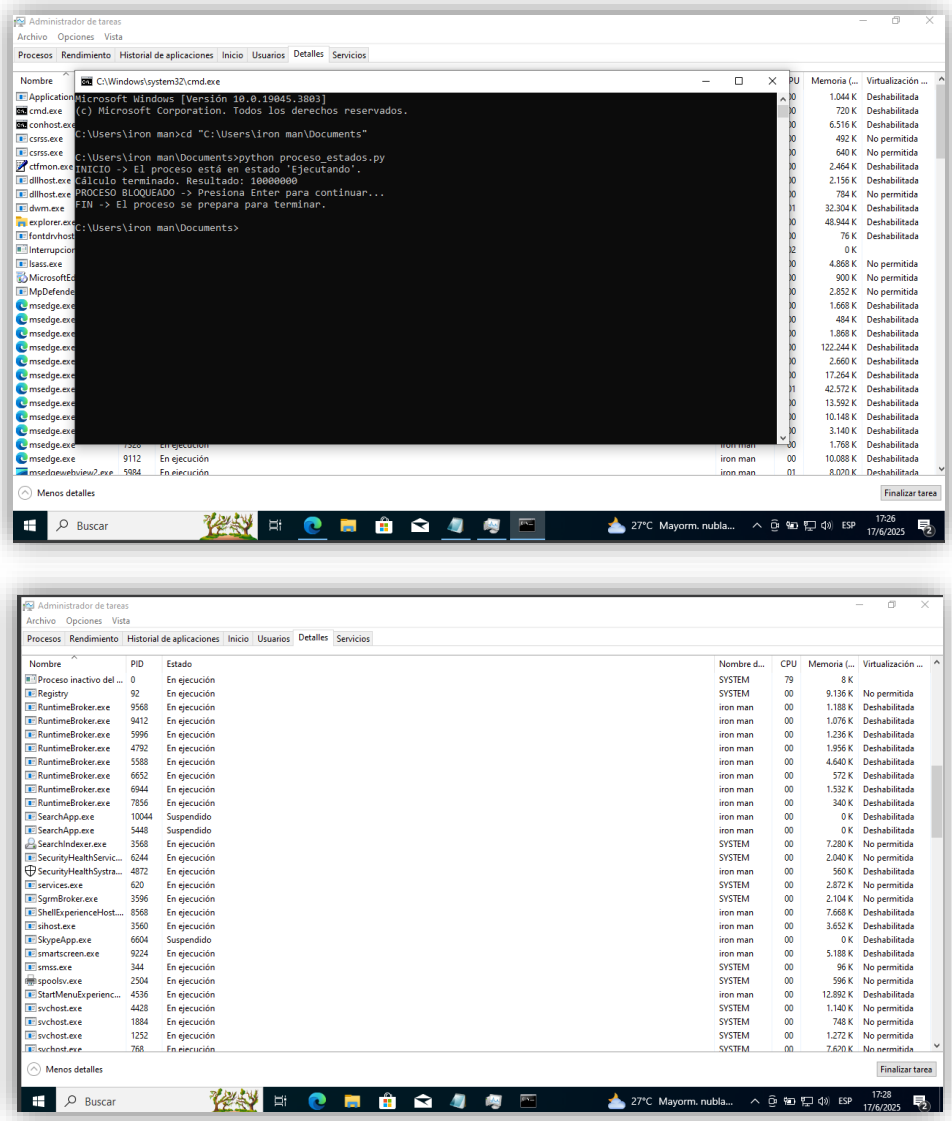

Figura 4. El proceso ha finalizado y ya no figura en el sistema.

Scheduling del Sistema Operativo.

Se ejecutaron simultáneamente cinco procesos intensivos en CPU (cpu_eater.py) para observar cómo el planificador distribuía el tiempo de procesador.

Observación: Como se evidencia en la Figura 5, el Administrador de Tareas mostró que los cinco procesos python.exe recibían porciones de tiempo de CPU de manera equitativa. Ningún proceso monopolizó la CPU por completo, sino que el sistema operativo alternaba rápidamente entre ellos.

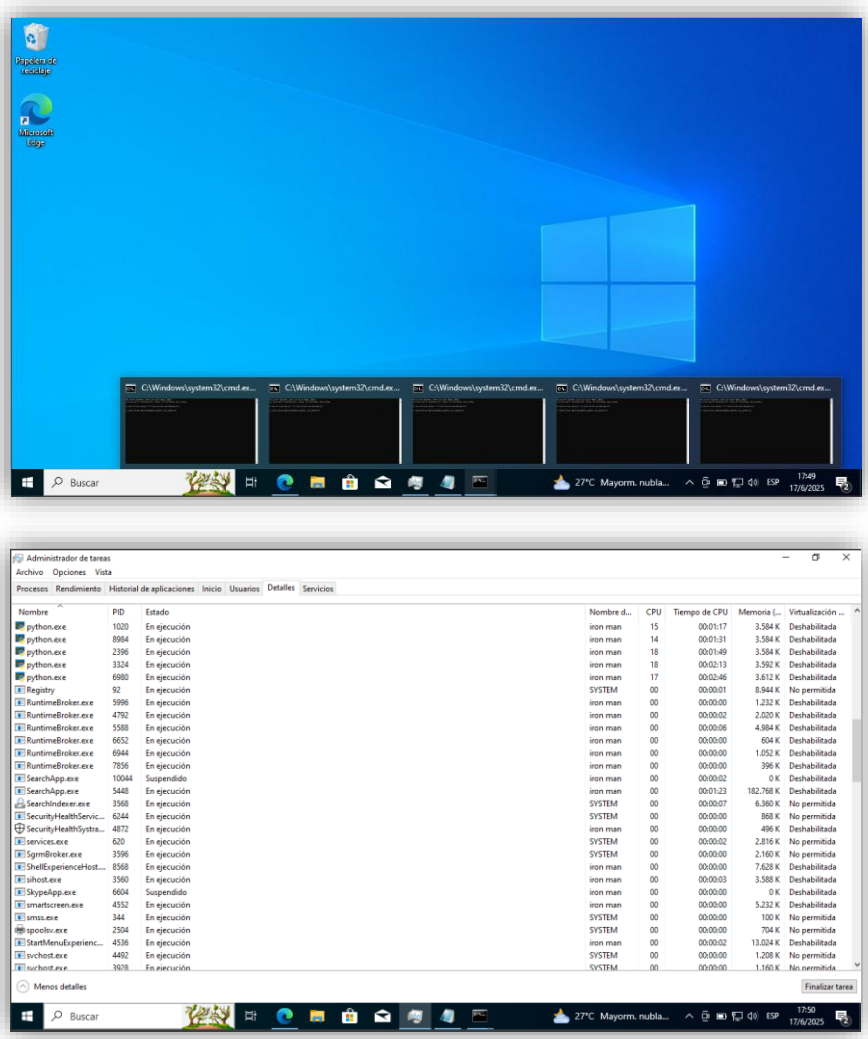

Figura 5. Distribución de tiempo de CPU entre 5 procesos competidores.

Análisis: Este comportamiento es consistente con un algoritmo de planificación apropiativo por turnos (similar a Round Robin), y no con un sistema FIFO (First-In, First-Out). Windows utiliza un sistema de planificación complejo basado en prioridades y turnos para garantizar la responsabilidad del sistema.

Simulación del Deadlock.

Se investigó y simuló un interbloqueo mediante dos scripts (proceso_A.py y proceso_B.py) que intentaban adquirir dos recursos (archivos de texto) en orden inverso.

Creación del Deadlock: Al ejecutar ambos scripts simultáneamente, cada uno logró bloquear un recurso y se quedó esperando indefinidamente por el recurso que el otro poseía. Ambas terminales quedaron congeladas, demostrando un deadlock exitoso.

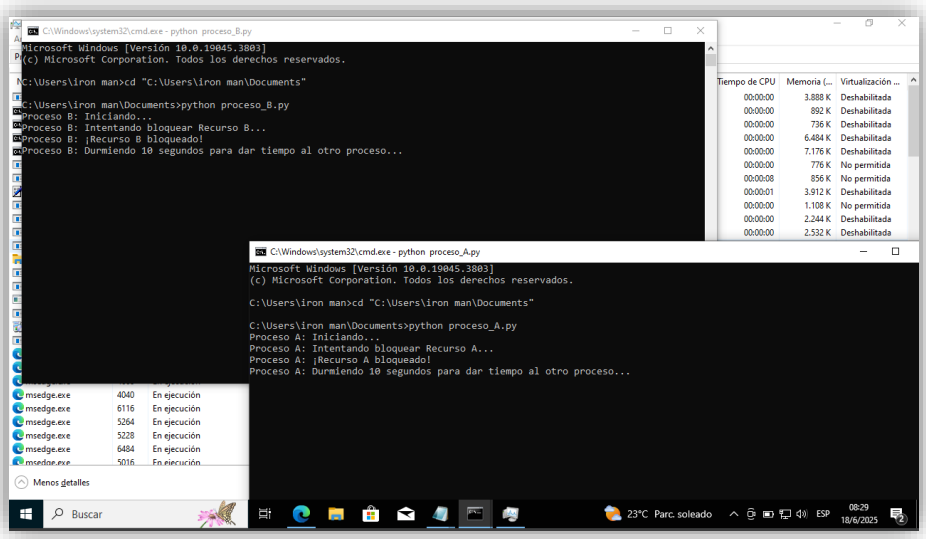

Figura 6. Deadlock simulado. Ambos procesos en espera mutua.

Resolución del Deadlock: El interbloqueo se resolvió de manera forzada finalizando uno de los procesos (python.exe) desde el Administrador de Tareas. Esto liberó el recurso que dicho proceso retenía, permitiendo que el segundo proceso completara su ejecución.

Conclusión.

Este laboratorio permitió verificar de forma práctica y controlada los conceptos teóricos de la gestión de procesos. Se logró documentar con éxito los cinco estados de un proceso, se observó el funcionamiento del planificador de Windows en un escenario de alta demanda y se demostró cómo puede ocurrir y resolverse un interbloqueo. Los resultados confirman el rol fundamental del sistema operativo como un gestor de recursos complejo y eficiente.