

Nombre: Luana Carolina Espinola Rivarola

Asignatura: Sistemas Operativos

Laboratorio 1: Gestión de Procesos.

Estados de los Procesos

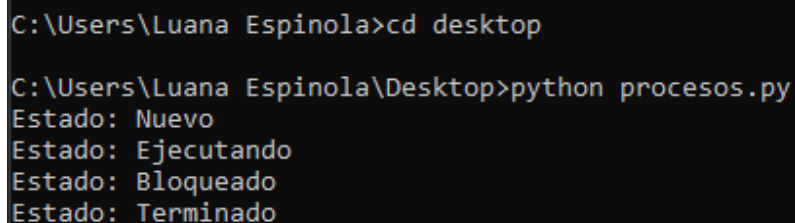
Descripción del Experimento

Se creó un programa en Python que simula los estados más comunes de un proceso en el sistema operativo: Nuevo, Listo, Ejecutando, Bloqueado y Terminado.

Análisis

Para este laboratorio se ejecutó un archivo llamado procesos.py, el cual simula el comportamiento de un proceso a lo largo del tiempo. Durante la ejecución se observaron los cambios en el Administrador de tareas, donde el proceso aparecía y desaparecía según lo que hacía el programa.

Con pausas intencionales, se logró visualizar cómo el proceso permanecía activo unos momentos y luego finalizaba.



```
C:\Users\Luana Espinola>cd desktop  
C:\Users\Luana Espinola\Desktop>python procesos.py  
Estado: Nuevo  
Estado: Ejecutando  
Estado: Bloqueado  
Estado: Terminado
```

Ilustración 1 Transición entre estados.

Para complementar la observación de los estados del proceso, modifiqué el archivo `procesos.py` para que registre el tiempo que pasa entre cada etapa del programa.

Con eso, pude saber cuántos segundos tarda desde que el proceso está listo hasta que se ejecuta, desde que está ejecutando hasta que queda en espera, y finalmente hasta que termina.

Esos tiempos los anoté en una tabla y los guardé en un archivo Excel llamado `datos_mediciones.xlsx`, que forma parte del repositorio.

```
C:\Users\Luana Espinola\Desktop>python procesos_tiempo.py
Estado: Nuevo
Estado: Ejecutando
Estado: Bloqueado
Estado: Terminado

--- Tiempos de transición ---
Listo a Ejecutando: 2.0 segundos
Ejecutando a Bloqueado: 3.0 segundos
Bloqueado a Terminado: 5.0 segundos
```

Ilustración 2 Tiempo de Transición entre estados en el cmd.

Transición	Tiempo (segundos)
Listo → Ejecutando	2
Ejecutando → Bloqueado	3
Bloqueado → Terminado	5

Ilustración 3 Excel de tiempo de transición entre procesos

Scheduling de un Sistema Operativo (Windows)

Descripción del experimento.

Se utilizaron comandos desde el símbolo del sistema (CMD) para abrir cinco instancias del mismo programa al mismo tiempo. Luego, se accedió al Administrador de tareas para observar cómo el sistema operativo repartía el uso del CPU entre los procesos.

Análisis

En la prueba se ejecutaron varias instancias del mismo programa desde el símbolo del sistema (CMD), todas con una alta demanda de CPU. En el Administrador de tareas se observa que el sistema reparte el uso del procesador entre los procesos, pero no de forma perfectamente igual. Los valores registrados fueron: 17.0%, 16.8%, 15.5%, 15.5% y 15.2%.

Esto indica que el sistema operativo intenta distribuir los recursos de manera equilibrada, aunque hay pequeñas variaciones naturales en el reparto. Estas diferencias pueden deberse a factores como el momento exacto en que se inicia cada proceso, pequeñas diferencias en ejecución interna o el orden en el que el planificador del sistema les da acceso al CPU.

Aunque no es un reparto idéntico, el comportamiento general se acerca a una planificación tipo Round Robin, donde los procesos reciben turnos similares para usar el procesador.

```

C:\> Símbolo del sistema - python cpu_intensivo.py

Microsoft Windows [Versión 10.0.19044.2604]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Luana Espinola>cd desktop

C:\Users\Luana Espinola\Desktop>python cpu_intensivo.py

```

Ilustración 4 Simulación de procesos pesados en CMD.

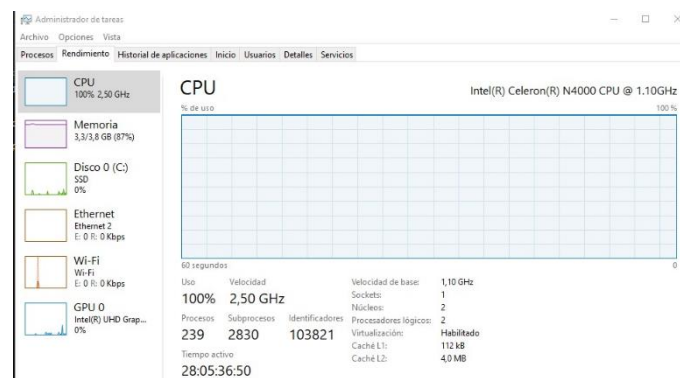


Ilustración 5 CPU del computador en 100% de uso

Administrador de tareas										
Procesos										
Nombre	Estado	100% CPU	86% Memoria	1% Disco	0% Red	0% GPU	Motor de la GPU	Consumo de ...	Tendencia de ...	
Procesador de comandos de Wi...		17,0%	4,2 MB	0 MB/s	0 Mbps	0%		Moderado	Bajo	
Procesador de comandos de Wi...		16,8%	4,2 MB	0 MB/s	0 Mbps	0%		Moderado	Bajo	
Procesador de comandos de Wi...		15,5%	10,9 MB	0 MB/s	0 Mbps	0%		Moderado	Bajo	
Procesador de comandos de Wi...		15,5%	10,9 MB	0 MB/s	0 Mbps	0%		Moderado	Bajo	
Procesador de comandos de Wi...		15,2%	10,8 MB	0 MB/s	0 Mbps	0%		Moderado	Bajo	
Administrador de tareas		3,7%	25,9 MB	0 MB/s	0 Mbps	0%		Muy baja	Muy baja	

Ilustración 6 Administrador de tarea repartiendo el uso de la CPU.

Creación de un Deadlock

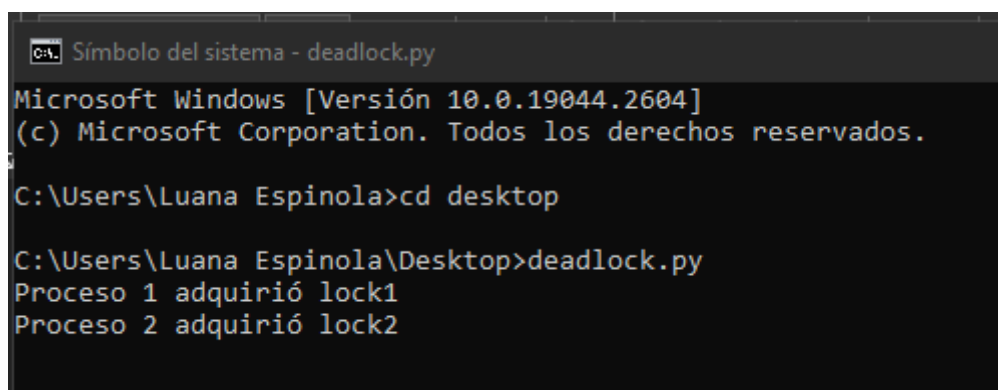
Descripción del experimento

Para esta parte del laboratorio, se utilizó un programa en Python diseñado para simular una situación de bloqueo entre dos procesos. El código consistía en dos funciones que intentaban acceder a dos recursos (bloqueos) en diferente orden, lo que genera una espera cruzada entre ambos.

El archivo fue ejecutado desde la consola de Windows. Durante su ejecución, se observó que los mensajes del programa se detenían en cierto punto, lo que indicaba que los procesos estaban esperando mutuamente sin poder avanzar.

Análisis final

El experimento permitió reproducir un deadlock de forma controlada. Una vez en estado de bloqueo, ninguno de los procesos logró finalizar su tarea, ya que cada uno esperaba a que el otro liberara el recurso. El sistema no intervino automáticamente, y fue necesario cerrar el programa de forma manual.

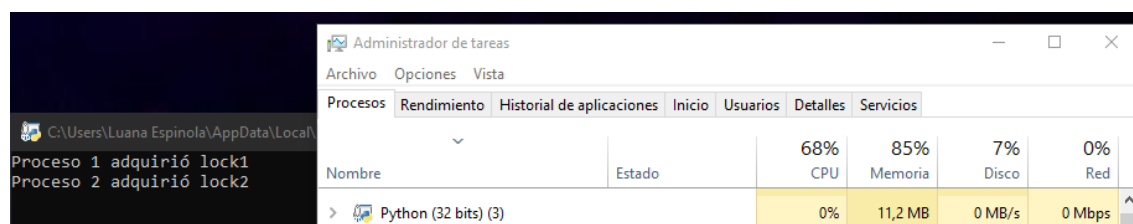


```

C:\Users\Luana Espinola>cd desktop

C:\Users\Luana Espinola\Desktop>deadlock.py
Proceso 1 adquirió lock1
Proceso 2 adquirió lock2
  
```

Ilustración 7 Código deadlock en CMD



Administrador de tareas				
Archivos Opciones Vista				
Procesos Rendimiento Historial de aplicaciones Inicio Usuarios Detalles Servicios				
Nombre	Estado	68% CPU	85% Memoria	7% Disco
Python (32 bits) (3)		0%	11,2 MB	0 MB/s

Ilustración 8 Procesos de Python haciendo deadlock

Para evitar el deadlock, se modificó el experimento haciendo que ambos procesos solicitaran los recursos en el mismo orden.

Después de ese cambio, el programa se ejecutó sin bloquearse y los procesos finalizaron correctamente.

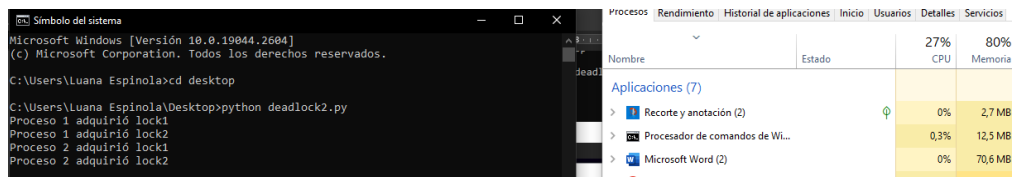


Ilustración 9 Procesos deadlock ejecutándose tras el arreglo