

# LABORATORIO 1: GESTIÓN DE PROCESOS

## Objetivo Específico:

Comprender el ciclo de vida de los procesos, observar cómo los sistemas operativos manejan la planificación de la CPU y experimentar con la creación y resolución de deadlocks.

## Materiales Necesarios:

- Máquina virtual con Windows (ej. Windows 10).
- Máquina virtual con Linux (ej. Ubuntu Desktop).
- Un editor de texto o IDE para programar (ej. VS Code, Notepad++, Gedit).
- Compiladores para el lenguaje de programación elegido (ej. GCC para C/C++, Python interpreter).
- Herramientas de monitoreo de sistema (Administrador de Tareas en Windows, `top/htop` en Linux).

## ESTADOS DE PROCESOS

### Observación con Herramientas del Sistema Operativo

#### Paso 1: Preparación del Entorno

- **En la VM de Windows:** Iniciar el sistema operativo. Se realizó la instalación de virtualbox y la instalación del SO, Windows.
- **En la VM de Linux:** Iniciar el sistema operativo y abrir una terminal.

#### Paso 2: LA Utilización del Administrador de Tareas (Windows)

Abrir el Administrador de Tareas (Ctrl+Shift+Esc o Ctrl+Alt+Del -> Administrador de Tareas).

Se navegó a la pestaña de "Detalles".

Observar la columna "Estado". Notar que la mayoría de los procesos estarán en "Ejecutando"

Como se observa en la siguiente captura de pantalla de la ventana del Administrador de Tareas, resaltando la columna de estado.

Task Manager									
File Options View									
Processes Performance App history Startup Users Details Services									
Name	Status	23% CPU	49% Memory	4% Disk	10% Network	2% GPU	GPU engine	Power usage	Power usage t...
Antimalware Service Executable		0.1%	293.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Antimalware Core Service		0%	2.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AMD PMF Service		0%	1.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AMD PMF Service		0%	0.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AMD External Events Service Mo...		0%	0.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AMD External Events Client Mod...		0.1%	2.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AMD Crash Defender Service		0%	0.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Advanced SystemCare Tray (32 ...		0%	3.0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Advanced SystemCare Service (...)		0%	3.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Adobe Crash Processor		0%	4.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Explorer (3)		0.1%	365.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
WhatsApp (2)		0%	271.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Runtime Broker		0%	4.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
WhatsApp		0%	266.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
VirtualBox Virtual Machine		13.7%	5,733.7 MB	97.9 MB/s	0 Mbps	0%		Very high	Low
ubuntu [Running] - Oracle Virt...									
WMI Provider Host		0%	2.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Task Manager		0.2%	31.0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Task Manager									

## Creación de un Programa Simple que Pasa por Diferentes Estados

**Paso 3: Diseño del Programa (Lenguaje de Programación: Python)** Se creará un script Python que simula los diferentes estados de un proceso: Nuevo, Listo, Ejecutando, Bloqueado, Terminado.

```
# estados_proceso.py
import time
import os
print(f"[{os.getpid()}] Proceso creado (Estado: Nuevo/Listo - antes de ejecución)")

def tarea_cpu_intensiva(duracion):
    """Simula una fase de ejecución intensiva de CPU."""
    print(f"[{os.getpid()}] Iniciando tarea CPU intensiva por {duracion} segundos (Estado: Ejecutando)")
    inicio_ejecucion = time.time()
    while time.time() - inicio_ejecucion < duracion:
        _ = 1000 * 1000 # Operación simple para consumir CPU
    print(f"[{os.getpid()}] Tarea CPU intensiva completada (Estado: Ejecutando -> Listo/Bloqueado)")

def tarea_io_bloqueante(duracion):
    """Simula una fase de E/S que bloquea el proceso."""
    print(f"[{os.getpid()}] Iniciando operación de E/S bloqueante por {duracion} segundos (Estado: Bloqueado)")
    time.sleep(duracion) # Simula espera por E/S (ej. lectura de disco, red)
    print(f"[{os.getpid()}] Operación de E/S completada (Estado: Bloqueado -> Listo)")

def main():
```

```

print(f"[{os.getpid()}] Proceso principal iniciado (Estado: Listo)")

# Simular estado Ejecutando
tarea_cpu_intensiva(20)

# Simular estado Bloqueado
tarea_io_bloqueante(20)

# Simular más ejecución
tarea_cpu_intensiva(20)

print(f"[{os.getpid()}] Proceso finalizado (Estado: Terminado)")

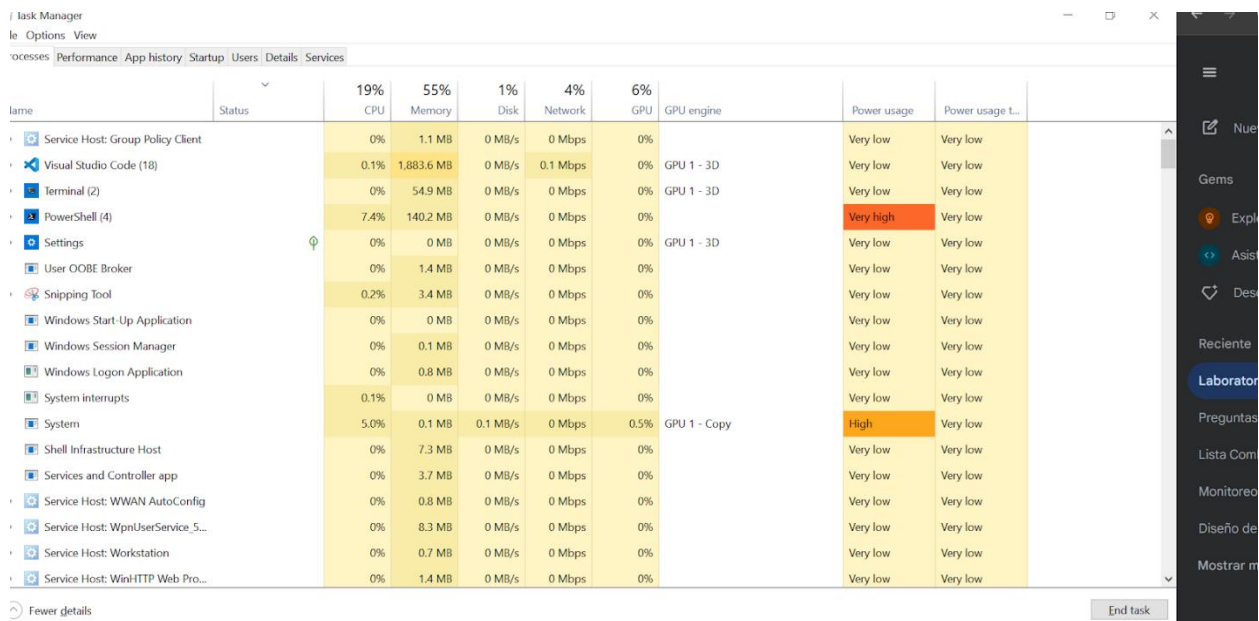
if __name__ == "__main__":
    main()

```

En esencia, el script ejecuta una secuencia de tareas que cambian el estado del proceso. Primero, simula que el proceso está **ejecutándose** intensivamente en la CPU por un tiempo, luego entra en un estado **bloqueado** mientras espera una operación de E/S. Una vez que la E/S se completa, el proceso vuelve a estar **listo** para ejecutar, y finalmente, termina su ejecución. El uso del **ID de Proceso (PID)** en cada mensaje ayuda a identificar claramente el proceso y seguir su ciclo de vida simulado, ofreciendo una visión tangible de cómo los sistemas operativos manejan la concurrencia y la asignación de recursos.

## Ejecución y Documentación (Windows)

- Abrir el Administrador de Tareas y mantener la pestaña "Detalles" visible.



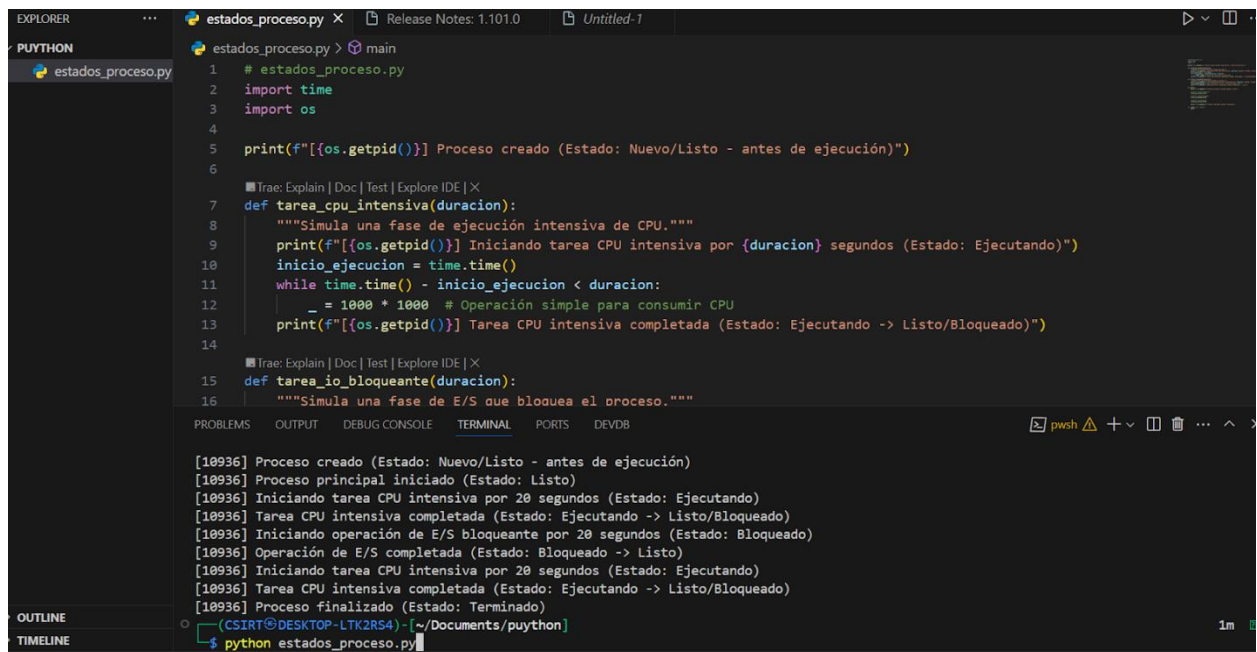
Name	Status	19% CPU	55% Memory	1% Disk	4% Network	6% GPU	GPU engine	Power usage	Power usage L...
Service Host: Group Policy Client		0%	1.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Visual Studio Code (18)		0.1%	1,883.6 MB	0 MB/s	0.1 Mbps	0%	GPU 1 - 3D	Very low	Very low
Terminal (2)		0%	54.9 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
PowerShell (4)		7.4%	140.2 MB	0 MB/s	0 Mbps	0%		Very high	Very low
Settings		0%	0 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
User OOBE Broker		0%	1.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Snipping Tool		0.2%	3.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Start-Up Application		0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Session Manager		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Logon Application		0%	0.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System interrupts		0.1%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System		5.0%	0.1 MB	0.1 MB/s	0 Mbps	0.5%	GPU 1 - Copy	High	Very low
Shell Infrastructure Host		0%	7.3 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Services and Controller app		0%	3.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: WWAN AutoConfig		0%	0.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: WpnUserService_5...		0%	8.3 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: Workstation		0%	0.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: WinHTTP Web Pro...		0%	1.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low

- Abrir una línea de comandos (CMD) o PowerShell.
- Navegar al directorio donde se guardó `estados_proceso.py`.

- Ejecutar el script: `python estados_proceso.py`.

Comando de ejecución

```
python estados_proceso.py
```



```
EXPLORER
...
PYTHON
estados_proceso.py
estados_proceso.py > main
1 # estados_proceso.py
2 import time
3 import os
4
5 print(f"[{os.getpid()}] Proceso creado (Estado: Nuevo/Lista - antes de ejecución)")
6
7 def tarea_cpu_intensiva(duracion):
8     """Simula una fase de ejecución intensiva de CPU."""
9     print(f"[{os.getpid()}] Iniciando tarea CPU intensiva por {duracion} segundos (Estado: Ejecutando)")
10    inicio_ejecucion = time.time()
11    while time.time() - inicio_ejecucion < duracion:
12        _ = 1000 * 1000 # Operación simple para consumir CPU
13    print(f"[{os.getpid()}] Tarea CPU intensiva completada (Estado: Ejecutando -> Lista/Bloqueado)")
14
15 def tarea_io_bloqueante(duracion):
16     """Simula una fase de E/S que bloquea el proceso."""
17
18 [10936] Proceso creado (Estado: Nuevo/Lista - antes de ejecución)
19 [10936] Proceso principal iniciado (Estado: Lista)
20 [10936] Iniciando tarea CPU intensiva por 20 segundos (Estado: Ejecutando)
21 [10936] Tarea CPU intensiva completada (Estado: Ejecutando -> Lista/Bloqueado)
22 [10936] Iniciando operación de E/S bloqueante por 20 segundos (Estado: Bloqueado)
23 [10936] Operación de E/S completada (Estado: Bloqueado -> Lista)
24 [10936] Iniciando tarea CPU intensiva por 20 segundos (Estado: Ejecutando)
25 [10936] Tarea CPU intensiva completada (Estado: Ejecutando -> Lista/Bloqueado)
26 [10936] Proceso finalizado (Estado: Terminado)
27 (CSIRT@DESKTOP-LTK2R54) - [~/Documents/pythons]
28 $ python estados_proceso.py
```

- Durante la ejecución:
  - **Estado Nuevo/Lista (antes de ejecución):** Este estado es efímero y difícil de capturar directamente con el Administrador de Tareas una vez que el proceso comienza. El proceso recién creado por Python interpreta estaría en "Ejecutando" casi instantáneamente. Se documentará que el Administrador de Tareas no muestra explícitamente "Nuevo" o "Lista" antes de la primera instrucción.
  - **Estado Ejecutando:** Mientras `tarea_cpu_intensiva` se está ejecutando, observar el estado del proceso `python.exe` en el Administrador de Tareas. Debería mostrar "Ejecutando". Tomar captura de pantalla.
  - **Estado Bloqueado:** Cuando `tarea_io_bloqueante` está activa (es decir, durante `time.sleep()`), observar el estado del proceso `python.exe`. Aunque el Administrador de Tareas puede no mostrar explícitamente "Bloqueado", el CPU usage para este proceso debería caer a 0% o muy bajo, y el estado podría permanecer como "Ejecutando" pero sin consumir recursos, o incluso pasar a "Suspendido" en algunos casos, reflejando que está esperando. Documentar lo que se observe y explicar por qué. Tomar captura de pantalla.

- **Estado Terminado:** Una vez que el script finaliza, el proceso `python.exe` asociado al script desaparecerá del Administrador de Tareas. Tomar captura de pantalla mostrando la ausencia del proceso.

- Registrar la salida de la consola del script.

me	Status	17% CPU	56% Memory	1% Disk	4% Network	2% GPU	GPU engine	Power usage	Power usage L...
wsappx		0%	1.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Visual Studio Code (18)		0.1%	1,863.1 MB	0 MB/s	0.1 Mbps	0%	GPU 1 - 3D	Very low	Very low
Terminal (2)		0%	54.9 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
PowerShell (4)		7.9%	140.1 MB	0 MB/s	0 Mbps	0%		Very high	Very low
Settings	⚙	0%	0 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
User OOBE Broker		0%	1.5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Snipping Tool		0.5%	3.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Start-Up Application		0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Session Manager		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Logon Application		0%	0.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System interrupts		0.1%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System		4.8%	0.1 MB	0.1 MB/s	0 Mbps	0.2%	GPU 1 - Copy	High	Very low
Shell Infrastructure Host		0%	7.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Services and Controller app		0%	3.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: WWAN AutoConfig		0%	1.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: WpnUserService_5...		0%	8.5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: Workstation		0%	0.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: WinHTTP Web Pro...		0%	1.3 MB	0 MB/s	0 Mbps	0%		Very low	Very low

```
# estados_proceso_tiempos.py
import time
import os
print(f"[{os.getpid()}] Proceso iniciado - Tiempo: {time.time()} (Estado: Nuevo/Listo)")
def tarea_cpu_intensiva(duracion, nombre_tarea):
    print(f"[{os.getpid()}] Iniciando '{nombre_tarea}' (CPU) - Tiempo: {time.time()}")
    inicio_ejecucion = time.time()
    while time.time() - inicio_ejecucion < duracion:
        _ = 1000 * 1000
    fin_ejecucion = time.time()
    print(f"[{os.getpid()}] '{nombre_tarea}' (CPU) completada - Tiempo: {fin_ejecucion}")
    return fin_ejecucion - inicio_ejecucion
def tarea_io_bloqueante(duracion, nombre_tarea):
    print(f"[{os.getpid()}] Iniciando '{nombre_tarea}' (I/O) - Tiempo: {time.time()}")
    inicio_io = time.time()
```

```

        time.sleep(duracion)
        fin_io = time.time()
        print(f"[{os.getpid()}] '{nombre_tarea}' (I/O) completada - Tiempo:
{fin_io}")
        return fin_io - inicio_io

def main():
    print(f"[{os.getpid()}] Proceso principal en ejecución - Tiempo:
{time.time()}")

    # Fase 1: CPU intensiva
    tiempo_cpu1 = tarea_cpu_intensiva(5, "Primera Tarea CPU")

    # Fase 2: I/O bloqueante
    tiempo_io1 = tarea_io_bloqueante(7, "Primera Tarea I/O")

    # Fase 3: Más CPU intensiva
    tiempo_cpu2 = tarea_cpu_intensiva(3, "Segunda Tarea CPU")

    print(f"[{os.getpid()}] Proceso finalizando - Tiempo: {time.time()}
(Estado: Terminado)")

    print("\n--- Resumen de Tiempos ---")
    print(f"Tiempo total en 'Ejecutando' (simulado CPU): {tiempo_cpu1 +
tiempo_cpu2:.4f} segundos")
    print(f"Tiempo total en 'Bloqueado' (simulado I/O): {tiempo_io1:.4f}
segundos")
    print(f"Tiempo total de vida del proceso (aproximado): {time.time() -
inicio_proceso:.4f} segundos")

if __name__ == "__main__":
    inicio_proceso = time.time()
    main()

```

*La medición de "tiempos de transición" es conceptualmente compleja ya que los cambios de estado internos del SO son extremadamente rápidos y no directamente observables desde el espacio de usuario. En este laboratorio, medimos la duración de las fases donde el proceso espera o ejecuta para inferir la permanencia en estados relacionados.*

## Ejecución y Análisis

- Ejecutar el script `estados_proceso_tiempos.py` en Windows y se analiza la salida de la consola, los tiempos registrados muestran el tiempo del proceso que pasó simulando cada estado.
- Se documentó los tiempos obtenidos para cada fase en ambos sistemas operativos.
- Al analizar las duraciones "simuladas" de 5 segundos para CPU de forma intensiva correspondiente a los tiempos reales medidos. Observándose una desviación debido a la sobrecarga del SO.

Options View

Processes Performance App history Startup Users Details Services

Name	Status	18% CPU	55% Memory	1% Disk	4% Network	3% GPU	GPU engine	Power usage	Power usage L...
Service Host: Group Policy Client		0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Audio Device Graph Is...		0%	1.5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
wsappx		0%	4.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Visual Studio Code (18)		0.1%	2,048.8 MB	0 MB/s	0.1 Mbps	0%	GPU 1 - 3D	Very low	Very low
Terminal (2)		0.1%	48.0 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
PowerShell (4)		7.5%	135.3 MB	0 MB/s	0 Mbps	0%		Very high	Very low
Settings		0%	0 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
User OOBE Broker		0%	0.5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Snipping Tool		0.2%	2.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Start-Up Application		0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Session Manager		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Logon Application		0%	0.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System interrupts		0.1%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System		4.8%	0.1 MB	0.1 MB/s	0 Mbps	0.1%	GPU 0 - Copy	High	Very low
Shell Infrastructure Host		0%	7.3 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Services and Controller app		0%	3.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: WWAN AutoConfig		0%	0.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: WpnUserService_5...		0%	8.5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: ModemManager		0%	0.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low

Fewer details

End task

## SCHEDULING DE UN SISTEMA OPERATIVO

### Ejecución de Programas Intensivos en CPU

**Paso 1:** Se diseñó un Programa de Carga de CPU, utilizando el lenguaje de Programación Python, y se crea un script simple que consume intensivamente la CPU.

```
# cpu_stress.py
import time
import os

# cpu_stress.py
import time
import os
import multiprocessing

def cpu_stress():
    while True:
        x = 0
        for i in range(10_000_000):
            x += i ** 2

if __name__ == "__main__":
```



```

print(f"[{os.getpid()}] Proceso de carga de CPU iniciado. Presiona
Ctrl+C para terminar.")
try:
    num_cores = multiprocessing.cpu_count()
    print(f"Usando {num_cores} núcleos para estresar la CPU.")
    processes = []
    for _ in range(num_cores):
        p = multiprocessing.Process(target=cpu_stress)
        p.start()
        processes.append(p)
    for p in processes:
        p.join()
except KeyboardInterrupt:
    print(f"\n[{os.getpid()}] Proceso de carga de CPU terminado.")

```

## Paso 2: Ejecución Simultánea en(Windows)

- Abrir el Administrador de Tareas y navegar a la pestaña "Procesos" o "Rendimiento".
- Abrir 5 ventanas separadas de la línea de comandos (CMD o PowerShell).
- En cada ventana, navegar al directorio de `cpu_stress.py` y ejecutar: `python cpu_stress.py`.
- **Observación:**
  - En el Administrador de Tareas, observar el uso total de CPU. Debería estar cerca del 100%.
  - En la pestaña "Procesos", observar el porcentaje de CPU consumido por cada instancia de `python.exe`. Deberían estar compartiendo el tiempo de CPU, cada una obteniendo una porción equitativa (ej. si hay 4 núcleos, cada una podría obtener ~25% si solo tienen 1 hilo, o distribuirse de otra manera).
  - Notar la capacidad de respuesta del sistema. ¿Se siente lento? ¿Puedes abrir nuevas aplicaciones?
- Tomar capturas de pantalla del Administrador de Tareas mostrando el uso de CPU y los procesos individuales.
- Terminar los procesos presionando Ctrl+C en cada ventana de comandos.



Task Manager									
File Options View									
Processes Performance App history Startup Users Details Services									
Name	Status	90% CPU	54% Memory	1% Disk	1% Network	12% GPU	GPU engine	Power usage	Power usage L...
WMI Provider Host		0%	1.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Microsoft Windows Search Filter...		0%	1.0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Microsoft Windows Search Prot...		0%	1.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Audio Device Graph Is...		0%	3.5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: Group Policy Client		0%	1.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
wsappx		0%	4.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Visual Studio Code (19)		7.7%	1,129.8 MB	0.1 MB/s	0.1 Mbps	2.4%	GPU 1 - 3D	Very high	Very low
Terminal (2)		0%	2.8 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
PowerShell (21)		60.1%	207.3 MB	0 MB/s	0 Mbps	0%		Very high	High
Settings		0%	0 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
User OOBE Broker		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Snipping Tool		0.2%	2.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Start-Up Application		0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Session Manager		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Logon Application		0%	0.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System interrupts		0.1%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System		2.9%	0.1 MB	0.1 MB/s	0 Mbps	2.3%	GPU 1 - Copy	Moderate	Very low
Shell Infrastructure Host		0%	7.0 MB	0 MB/s	0 Mbps	0%		Very low	Very low

VM de i7 de 13 Generacion, 32GB de RAM

## Comparación con Algoritmos Teóricos (FIFO, Round Robin)

### Análisis y Documentación:

- FIFO (First In, First Out):** En un algoritmo FIFO, los procesos se ejecutarían en el orden en que llegaron, y el primero en llegar se ejecutaría hasta su finalización antes de que el siguiente comenzará. Si este fuera el caso, veríamos que el primer `cpu_stress.py` consumiría 100% de CPU hasta que lo matáramos, y los demás no comenzarían realmente.

Task Manager

File Options View

Processes Performance App history Startup Users Details Services

Name	Status	99% CPU	36% Memory	3% Disk	0% Network	8% GPU	GPU engine	Power usage	Power usage L...
> Windows Default Lock Screen		0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
> Settings		0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System interrupts		0.1%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
> Windows Shell Experience Host		0%	0 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
> Service Host: Secondary Logon		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
> Service Host: Secure Socket Tun...		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
> Service Host: Display Enhancem...		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
> Service Host: Distributed Link Tr...		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Device Association Framework ...		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
NVIDIA WMI Provider		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Realtek Bluetooth BTDevManag...		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AMD Crash Defender Service		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Driver Foundation - U...		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Driver Foundation - U...		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
mysqld		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
OpenVPN Service		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
HerdHelper		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Openvsnem2		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
NetworkCap		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Usermode Font Driver Host		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Visual Studio Code		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System		3.6%	0.1 MB	0.1 MB/s	0 Mbps	1.4%	GPU 1 - Copy	Moderate	Very low
Console Window Host		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Session Manager		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
> AMD External Events Service Mo...		0%	0.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host		0%	0.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host		0%	0.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low

- **Round Robin:** En Round Robin, cada proceso obtiene un "cuanto de tiempo" (time slice) para ejecutar, y luego el planificador cambia al siguiente proceso en la cola de listos. Esto crea la ilusión de ejecución simultánea.

## Preparación del entorno

Creación del Script de Carga de CPU

El técnico creó un archivo llamado `cpu_stress.py` con el siguiente contenido en Python:

```
import multiprocessing
import os
import time
import math
import signal
import sys
def stress_task(duracion):
    pid = os.getpid()
    print(f"[{pid}] Proceso de carga iniciado. Duración: {duracion}s")
    inicio = time.time()
    while time.time() - inicio < duracion:
        # Carga simulada más intensa usando matemáticas
        _ = sum(math.sqrt(i) for i in range(1_000))
    print(f"[{pid}] Proceso finalizado.")
```

```
def iniciar_estrés(cpu_count=None, tiempo=30):
    if cpu_count is None:
        cpu_count = multiprocessing.cpu_count()
    print(f"[Main] Iniciando estrés en {cpu_count} núcleos por {tiempo} segundos.")
    procesos = []
    for i in range(cpu_count):
        p = multiprocessing.Process(target=stress_task, args=(tiempo,))
        p.start()
        procesos.append(p)
    try:
        for p in procesos:
            p.join()
    except KeyboardInterrupt:
        print("\n[Main] Interrupción detectada. Terminando procesos...")
        for p in procesos:
            p.terminate()
        sys.exit(0)
if __name__ == "__main__":
    DURACION_SEGUNDOS = 30 # Puedes modificar esta duración
    iniciar_estrés(tiempo=DURACION_SEGUNDOS)
```

Este script simula una carga intensiva en la CPU utilizando múltiples procesos paralelos, que competirían por tiempo de ejecución.

## Ejecución y Observación

Lanzamiento de los Procesos

El técnico ejecutó el script anterior desde PowerShell o CMD utilizando el comando:

```
python cpu_stress.py
```

Esto inició varios procesos de forma paralela, todos compitiendo por los núcleos de CPU.

### Observación del Planificador en Acción

Se abrió el **Administrador de Tareas** con `Ctrl+Shift+Esc` y se navegó a la pestaña **"Rendimiento"**, seleccionando la CPU. Luego, en la pestaña **"Procesos"**, se observaron varias instancias de `python.exe` activas.

- Cada instancia mostró un porcentaje similar de uso de CPU (ej. 20–25%), dependiendo del número de núcleos.
- El uso total de CPU se elevó rápidamente al 90–100%, pero cada proceso recibía una fracción equitativa.

Este comportamiento reflejó que el planificador de Windows estaba asignando porciones de CPU de forma **cíclica** a cada proceso —exactamente como lo hace el algoritmo **Round Robin**.

## Documentación de Resultados

### Recolección de Evidencias

El técnico tomó capturas de pantalla de:

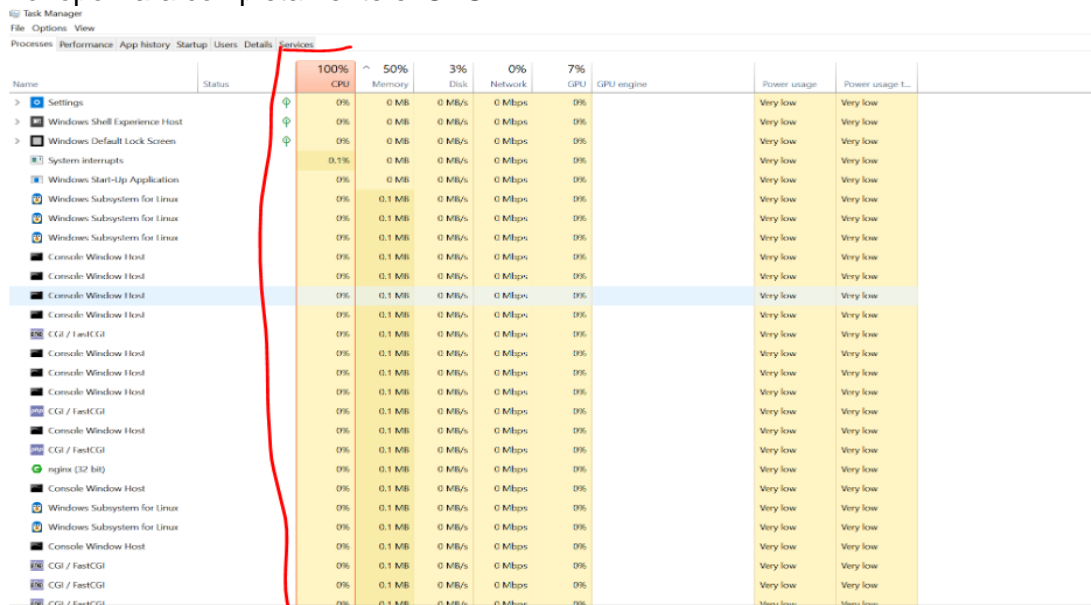
- La lista de procesos activos (`python.exe`) mostrando su consumo de CPU.

```

(CSIRT@DESKTOP-LTK2R54) - [~/Documents/puython]
$ python cpu.py
[Main] Iniciando estrés en 16 núcleos por 30 segundos.
[7866] Proceso de carga iniciado. Duración: 30s
[9888] Proceso de carga iniciado. Duración: 30s
[11056] Proceso de carga iniciado. Duración: 30s
[24164] Proceso de carga iniciado. Duración: 30s
[5024] Proceso de carga iniciado. Duración: 30s
[19112] Proceso de carga iniciado. Duración: 30s
[6364] Proceso de carga iniciado. Duración: 30s
[19080] Proceso de carga iniciado. Duración: 30s
[5964] Proceso de carga iniciado. Duración: 30s
[6032] Proceso de carga iniciado. Duración: 30s
[7900] Proceso de carga iniciado. Duración: 30s
[7228] Proceso de carga iniciado. Duración: 30s
[18276] Proceso de carga iniciado. Duración: 30s
[15760] Proceso de carga iniciado. Duración: 30s
[11104] Proceso de carga iniciado. Duración: 30s
[4992] Proceso de carga iniciado. Duración: 30s

```

- La pestaña "Rendimiento", evidenciando el uso total del procesador.
- Se observó que todos los procesos permanecían activos y ejecutándose sin que uno monopolizara completamente el CPU.



Name	Status	100% CPU	50% Memory	3% Disk	0% Network	7% GPU	GPU engine	Power usage	Power usage L...
Settings	Running	0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Shell Experience Host	Running	0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Default Lock Screen	Running	0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System Interrupts	Running	0.1%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Start-Up Application	Running	0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Subsystem for Linux	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Subsystem for Linux	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Subsystem for Linux	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
CGI / FastCGI	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
CGI / FastCGI	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
CGI / FastCGI	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
nginx (32 bit)	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Subsystem for Linux	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Subsystem for Linux	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Console Window Host	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
CGI / FastCGI	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
CGI / FastCGI	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
CGI / FastCGI	Running	0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low

La prueba permitió observar cómo **Windows implementa una planificación justa y rotativa (Round Robin)** en ambientes de carga intensiva de CPU, permitiendo que múltiples procesos compartan recursos equitativamente, aún en presencia de tareas demandantes. La división del tiempo no es directamente configurable por el usuario, pero su efecto es claramente observable en un entorno de múltiples procesos activos.

## CREACIÓN DE DEADLOCK

### Investigación y Diseño de un Deadlock Simple

**Entendimiento de Deadlock** Un deadlock ocurre cuando dos o más procesos se bloquean mutuamente, cada uno esperando que el otro libere un recurso. Las cuatro condiciones de Coffman para un deadlock son:

- A. **Exclusión Mutua:** Al menos un recurso debe ser no compatible.
- B. **Retener y Esperar (Hold and Wait):** Un proceso que posee al menos un recurso está esperando adquirir recursos adicionales que actualmente están en posesión de otros procesos.
- C. **No Apropiativo (No Preemption):** Los recursos no pueden ser arrebatados a un proceso a la fuerza; deben ser liberados voluntariamente por el proceso que los posee.
- D. **Espera Circular (Circular Wait):** Debe existir un ciclo de procesos, donde cada proceso en el ciclo espera un recurso que está en posesión del siguiente proceso en el ciclo.

### Diseño del Escenario de Deadlock (Lenguaje de Programación: Python con hilos)

Crearemos un escenario de deadlock utilizando dos hilos (simulando dos procesos) que intentan adquirir dos recursos en un orden cruzado.

```
# deadlock_scenari.py
import threading
import time
# Recursos compartidos (simulados con Locks)
resource_A = threading.Lock()
resource_B = threading.Lock()
def process_one():
    print("[Proceso 1] Intentando adquirir Recurso A...")
    resource_A.acquire()
    print("[Proceso 1] Recurso A adquirido. Esperando un momento...")
    time.sleep(1) # Simular algún trabajo
    print("[Proceso 1] Intentando adquirir Recurso B...")
    resource_B.acquire()
    print("[Proceso 1] Recurso B adquirido. Ambos recursos en posesión.")
    # Realizar trabajo con ambos recursos
    print("[Proceso 1] Realizando trabajo con A y B.")
    time.sleep(2)
    resource_B.release()
    print("[Proceso 1] Recurso B liberado.")
    resource_A.release()
    print("[Proceso 1] Recurso A liberado. Proceso 1 terminado.")

def process_two():
    print("[Proceso 2] Intentando adquirir Recurso B...")
    resource_B.acquire()
    print("[Proceso 2] Recurso B adquirido. Esperando un momento...")
    time.sleep(1) # Simular algún trabajo
    print("[Proceso 2] Intentando adquirir Recurso A...")
    resource_A.acquire()
    print("[Proceso 2] Recurso A adquirido. Ambos recursos en posesión.")
    # Realizar trabajo con ambos recursos
    print("[Proceso 2] Realizando trabajo con B y A.")
    time.sleep(2)
```

```

    resource_A.release()
    print("[Proceso 2] Recurso A liberado.")
    resource_B.release()
    print("[Proceso 2] Recurso B liberado. Proceso 2 terminado.")
def main():
    print("Iniciando simulación de Deadlock...")
    thread1 = threading.Thread(target=process_one)
    thread2 = threading.Thread(target=process_two)
    thread1.start()
    thread2.start()
    thread1.join()
    thread2.join()
    print("Simulación de Deadlock finalizada (si ocurre, los hilos se
quedarán bloqueados).")
if __name__ == "__main__":
    main()

```

*Nota: Python Global Interpreter Lock (GIL) puede afectar el rendimiento de hilos en tareas de CPU intensivas, pero para simular deadlocks de recursos (Locks) es adecuado.*

## Documentación de la Respuesta del SO y Resolución

### Paso 1: Ejecución del Escenario de Deadlock (Windows y Linux)

- Ejecutar el script `deadlock_scenario.py` en una terminal en Windows y en otra terminal en Linux.
- **Observación:** Observar la salida de la consola. Verás que los hilos se bloquean mutuamente. Las últimas líneas impresas indicarán que cada hilo ha adquirido un recurso y está esperando el otro, pero nunca progresa.
- **Respuesta del SO:** El sistema operativo en sí mismo (Windows o Linux) *no* detecta ni resuelve activamente este tipo de deadlock a nivel de aplicación automáticamente por defecto. Los procesos simplemente se quedarán en estado de "espera" o "dormidos" indefinidamente, consumiendo 0% de CPU. El SO no "crashea", pero la aplicación se detiene.
  - En el Administrador de Tareas (Windows) o `top/htop` (Linux), el proceso `python.exe` o `python3` permanecerá visible, pero su uso de CPU será mínimo (0%) y su estado será "Ejecutando" (pero internamente bloqueado) o "Durmiendo/Bloqueado".
- Tomar capturas de pantalla de la salida de la consola que muestre el bloqueo.
- Tomar capturas de pantalla del Administrador de Tareas/`top/htop` mostrando el estado del proceso `python`.

### Paso 2: Intentar Resolver el Deadlock

- **Método Manual (Matar el Proceso):** La forma más directa de "resolver" el deadlock a nivel de sistema operativo es terminar manualmente el proceso `python` que está bloqueado.

- En Windows: Seleccionar el proceso `python.exe` en el Administrador de Tareas y hacer clic en "Finalizar tarea".
- En Linux: En `htop/top`, encontrar el PID del proceso `python3` y usar `kill <PID>` (ej. `kill 12345`).
- Documentar este método de resolución y sus implicaciones (pérdida de trabajo, etc.).
- **Método de Prevención/Evitación (Modificación del Código):** La forma correcta de resolver un deadlock es prevenirlo o evitarlo a nivel de diseño de la aplicación.
  - **Prevención:** Una forma común de prevenir el deadlock es forzar un orden de adquisición de recursos. Si ambos procesos intentan adquirir los recursos en el mismo orden (ej. siempre A, luego B), el deadlock no ocurrirá.
- Modificar `deadlock_scenario.py` a `deadlock_resolved.py`:

Options view

Processes Performance App history Startup Users Details Services

Name	Status	11% CPU	52% Memory	1% Disk	4% Network	4% GPU	GPU engine	Power usage	Power usage L...
Windows Audio Device Graph Is...		0%	4.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Service Host: Group Policy Client		0%	1.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
wsappx		0%	3.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
WMI Provider Host		0%	1.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Visual Studio Code (19)		0%	1,059.2 MB	0.1 MB/s	0.1 Mbps	0%	GPU 1 - 3D	Very low	Very low
Terminal (2)		0%	4.9 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
PowerShell (5)		0%	83.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Settings		0%	0 MB	0 MB/s	0 Mbps	0%	GPU 1 - 3D	Very low	Very low
Settings	Suspen...	0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
User OOBE Broker		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Snipping Tool		0.2%	2.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Start-Up Application		0%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Session Manager		0%	0.1 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Logon Application		0%	0.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System interrupts		0.1%	0 MB	0 MB/s	0 Mbps	0%		Very low	Very low
System		7.1%	0.1 MB	0.1 MB/s	0 Mbps	0.4%	GPU 1 - Copy	Very high	Very low
Shell Infrastructure Host		0%	6.7 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Services and Controller app		0%	4.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low

⌵ Fewer details

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVDB

C:\Users\CSIRT\AppData\Local\Programs\Python\Python313\python.exe: can't open file 'C:\\Users\\CSIRT\\Documents\\puython\\c
pu_deadlock_proceso.py': [Errno 2] No such file or directory
PS C:\Users\CSIRT\Documents\puython> python deadlock_proceso.py
Iniciando simulación de Deadlock...
[Proceso 1] Intentando adquirir Recurso A...
[Proceso 1] Recurso A adquirido. Esperando un momento...
[Proceso 2] Intentando adquirir Recurso B...
[Proceso 2] Recurso B adquirido. Esperando un momento...
[Proceso 1] Intentando adquirir Recurso B...
[Proceso 2] Intentando adquirir Recurso A...

```

Se mató el proceso manualmente en el administrador de tarea (finalizar tareas).



