

LABORATORIO 1:

“GESTION DE PROCESOS”

ESTUDIANTE: WILLIAN PAREDES RAMIREZ

CARRERA: ING. INFORMATICA

2.025

#Laboratorio 1: Gestion de Procesos.

#OBJETIVO:

- Utilizar el top/htop (Linux).

```
bash
1 htop
```

- Creacion de un programa simple que pase por diferentes estados, Ubuntu contiene el lenguaje python3, luego, se abre el editor de texto: ejemplo:

```
bash
1 nano estado_proceso.py
```

```
import time
import os

print(f"[PID: {os.getpid()}] Estado NUEVO: Proceso creado")
input("Presiona Enter para pasar a LISTO...")

print("[EJECUTANDO] Usando CPU...")
start = time.time()
while time.time() - start < 20:
    sum(range(100000)) # Uso intensivo de CPU

print("[BLOQUEADO] Durmiendo 10 segundos...")
time.sleep(10) # Simula espera por E/S o recurso

print("[TERMINADO]")
```

- Observar como un proceso pasa por diferentes estados (Nuevo, Listo, Ejecutando, Bloqueado, Terminado). En las capturas podemos observar la linea de proceso que se formo.
- Medicion de los tiempos de transicion entre estados. Cumpliendo los siguientes esquemas:
- **R (Running)**: En ejecución.
- **S (Sleeping)**: Esperando (como cuando usas `sleep()`).
- **Z (Zombie)**: Proceso terminado pero no liberado.

#METODOS:

- Se creó un script en python3 que simula un proceso pasando por diferentes estados.
- Se usó “htop” para monitorear el proceso en tiempo real.

#RESULTADOS:

- Capturas de pantalla de “htop” mostrando cada estado.
- Tiempo de transición entre estados.

#ANALISIS:

- Descripción de como el Sistema Operativo gestiona los estados de los procesos.



16 de jun 21:45

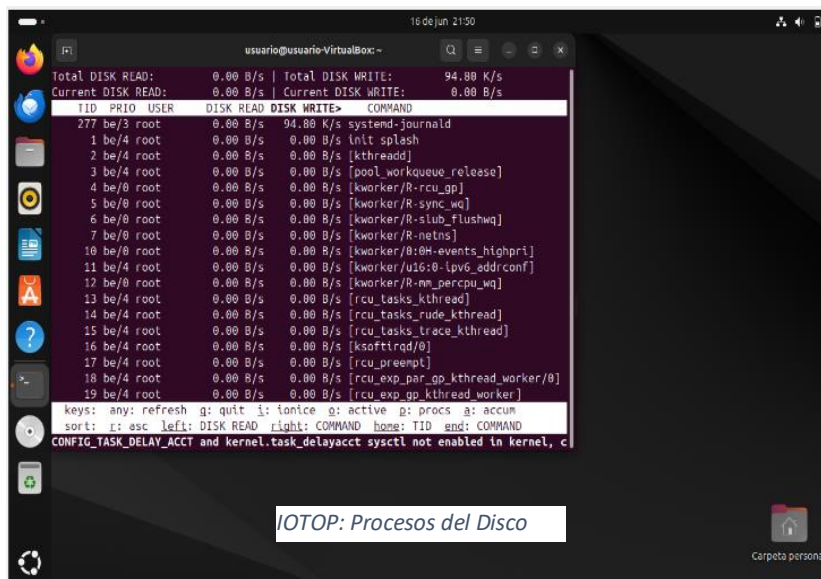
usuario@usuario-VirtualBox: ~

```

0[||||] 8.3% Tasks: 104, 339 thr, 109 kthr; 1 runn
1[||||] 6.9% Load average: 0.06 0.12 0.14
2[||||] 3.2% Uptime: 00:15:30
3[||||] 2.5%
Mem[|||||] 864M/7.75G
Swp[|] 0K/4.00G

Main 17.0
PID USER PRI NI VIRT RES SHR S CPU%MEM% TIME+ Command
1420 usuario 20 0 4820M 370M 138M S 8.9 4.7 0:12.71 /usr/bin/gnom
9727 usuario 20 0 19912 4736 3584 R 4.5 0.1 0:02.99 htop
1450 usuario -21 0 4820M 370M 138M S 1.3 4.7 0:01.12 /usr/bin/gnom
1453 usuario 20 0 4820M 370M 138M S 1.3 4.7 0:03.52 /usr/bin/gnom
1454 usuario 20 0 4820M 370M 138M S 1.3 4.7 0:03.54 /usr/bin/gnom
1452 usuario 20 0 4820M 370M 138M S 0.6 4.7 0:03.81 /usr/bin/gnom
1455 usuario 20 0 4820M 370M 138M S 0.6 4.7 0:03.91 /usr/bin/gnom
1468 usuario 20 0 4820M 370M 138M S 0.6 4.7 0:00.45 /usr/bin/gnom
1930 usuario 20 0 2081M 63696 47188 S 0.6 0.8 0:00.95 gjs /usr/shar
9235 usuario 20 0 552M 55416 43964 S 0.6 0.7 0:00.76 /usr/libexec/
1 root 20 0 23188 13924 9316 S 0.0 0.2 0:01.41 /sbin/init sp
277 root 19 -1 67096 17816 16536 S 0.0 0.2 0:00.28 /usr/lib/syst
350 root 20 0 30532 8540 4956 S 0.0 0.1 0:00.14 /usr/lib/syst
F1?help F2Setup F3search F4filter F5tree F6sortby F7Nice F8Nice F9Kill F10Quit
  
```

HTOP: ver en tiempo real la CPU, memoria y tiempo de ejecución



16 de jun 21:50

usuario@usuario-VirtualBox: ~

```

Total DISK READ: 0.00 B/s | Total DISK WRITE: 94.00 K/s
Current DISK READ: 0.00 B/s | Current DISK WRITE: 0.00 B/s

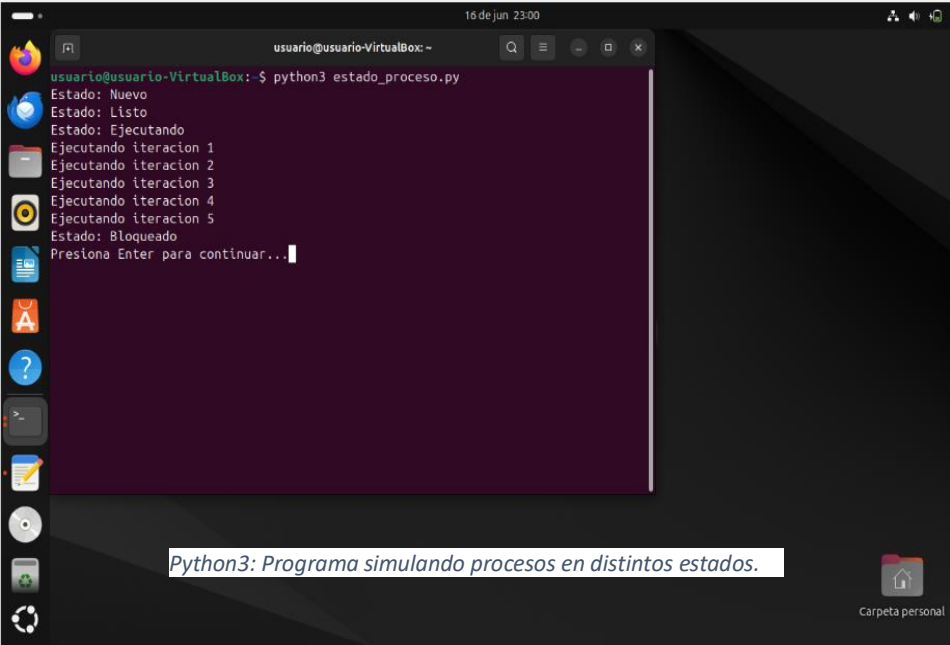
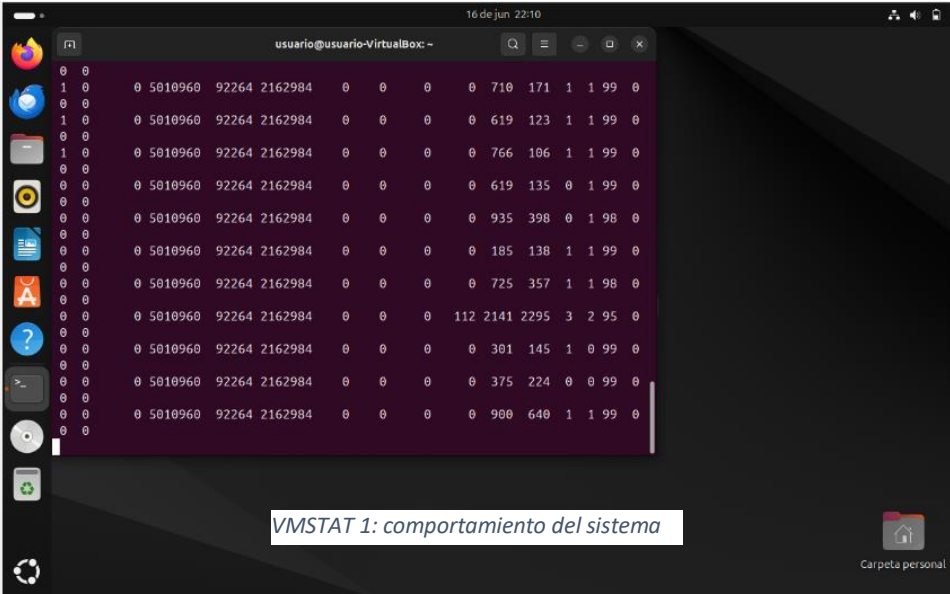
TID PRIO USER DISK READ DISK WRITE COMMAND
277 be/3 root 0.00 B/s 94.00 K/s systemd-journald
1 be/4 root 0.00 B/s 0.00 B/s init splash
2 be/4 root 0.00 B/s 0.00 B/s [kthreadd]
3 be/4 root 0.00 B/s 0.00 B/s [pool_workqueue_release]
4 be/0 root 0.00 B/s 0.00 B/s [kworker/R-rcu_gp]
5 be/0 root 0.00 B/s 0.00 B/s [kworker/R-sync_wq]
6 be/0 root 0.00 B/s 0.00 B/s [kworker/R-slub_flushwq]
7 be/0 root 0.00 B/s 0.00 B/s [kworker/R-netns]
10 be/0 root 0.00 B/s 0.00 B/s [kworker/0:0H-events_highpri]
11 be/4 root 0.00 B/s 0.00 B/s [kworker/u16:0-love_addrconf]
12 be/0 root 0.00 B/s 0.00 B/s [kworker/R-nm_percpu_wq]
13 be/4 root 0.00 B/s 0.00 B/s [rcu_tasks_kthread]
14 be/4 root 0.00 B/s 0.00 B/s [rcu_tasks_rude_kthread]
15 be/4 root 0.00 B/s 0.00 B/s [rcu_tasks_trace_kthread]
16 be/4 root 0.00 B/s 0.00 B/s [ksftfired/0]
17 be/4 root 0.00 B/s 0.00 B/s [rcu_preempt]
18 be/4 root 0.00 B/s 0.00 B/s [rcu_exp_par_gp_kthread_worker/0]
19 be/4 root 0.00 B/s 0.00 B/s [rcu_exp_gp_kthread_worker]

keys: any: refresh q: quit i: toggle g: active p: proc a: accum
sort: r: asc left: DISK READ right: COMMAND home: TID end: COMMAND
CONFIG_TASK_DELAY_ACCT and kernel.task_delayacct sysctl not enabled in kernel, c
  
```

IOTOP: Procesos del Disco



Universidad de la Integración de las Américas
PARAGUAY



PID	USUARIO		VIRTUAL	CPU	MEMORIA	TIEMPO	COMANDO.
2408	root	20	0	1949M	38420	24668 S	0.0 0.5 0:00.01 /usr/lib/snap

ACCESO A LA TERMINAL HTOP

2429	usuario	20	0	29120	9472	6528 S	0.0 0.1 0:00.01 python3 estad
------	---------	----	---	-------	------	--------	-------------------------------

PROCESO NUEVO

2408	root	20	0	1949M	38420	24668 S	0.0 0.5 0:00.01 /usr/lib/snap
------	------	----	---	-------	-------	---------	-------------------------------

PROCESO LISTO

2429	usuario	20	0	29120	9656	6712 S	0.0 0.1 0:20.00 python3 estad
------	---------	----	---	-------	------	--------	-------------------------------

PROCESO BLOQUEADO

2408	root	20	0	1949M	38420	24668 S	0.0 0.5 0:00.01 /usr/lib/snap
------	------	----	---	-------	-------	---------	-------------------------------

PROCESO TERMINADO

real	0m38,368s
user	0m20,010s
sys	0m0,005s

Real: 38 s. en total

En CPU: 20 s.

Llamadas al sistema: 0.005s.

#Scheduling de un Sistema Operativo:

#Objetivo:

- Ejecutar 5 programas intensivos simultáneamente y observar como el Sistema Operativo distribuye el tiempo de CPU.

#Métodos:

- Se crearon 5 estancias de un script python que consume CPU.

```
import time
import os

print(f"[CPU Intensive] PID: {os.getpid()} - Iniciando consumo de CPU...")
while True:
    start = time.time()
    while time.time() - start < 5:
        sum(range(1000000)) # Fixed indentation and logic
    time.sleep(1)
```

- Se usaron herramientas como “htop”, “top” y “sar” para monitorear el uso de CPU.

```
bash
1 sar -u 1 10
```

#Resultados:

- Gráficos o tablas mostrando el uso de CPU por proceso.
- Comparación con algoritmos teóricos (FIFO, Round Robin).

```
from collections import deque

def fifo(procesos):
    tiempo = 0
    espera_total = 0
    for p in procesos:
        espera_total += tiempo
        tiempo += p['duracion']
    return espera_total / len(procesos)

def round_robin(procesos, quantum=2):
    cola = deque(procesos.copy())
    tiempo = 0
    espera_total = 0
    while cola:
        p = cola.popleft()
        if p['duracion'] > quantum:
            tiempo += quantum
            p['duracion'] -= quantum
            cola.append(p)
        else:
            tiempo += p['duracion']
            espera_total += tiempo - p['duracion_original']
    return espera_total / len(procesos)
```

```
# Ejemplo de procesos: {nombre, duración, duración_original}
procesos = [
    {'nombre': 'P1', 'duracion': 5, 'duracion_original': 5},
    {'nombre': 'P2', 'duracion': 3, 'duracion_original': 3},
    {'nombre': 'P3', 'duracion': 8, 'duracion_original': 8},
]

print("FIFO - Tiempo promedio de espera:", fifo(procesos))
print("Round Robin - Tiempo promedio de espera:", round_robin(procesos))
```

```
usuario@usuario-VirtualBox:~$ python3 fifo_vs_roundrobin.py
FIFO - Tiempo promedio de espera: 4.333333333333333
Round Robin - Tiempo promedio de espera: 7.0
```

#Análisis:

- Interpretación de como el Sistema Operativo realiza la planificación de procesos.



```
usuario@usuario-VirtualBox: ~
usuario@usuario-VirtualBox:~$ for i in {1..5}; do
> python3 cpu_intensivo_ubuntu.py &
> done
[1] 2961
[2] 2962
[3] 2963
[4] 2964
[5] 2965
usuario@usuario-VirtualBox:~$ [CPU Intensive] PID: 2961 - Iniciando consumo de CPU...
[CPU Intensive] PID: 2962 - Iniciando consumo de CPU...
[CPU Intensive] PID: 2963 - Iniciando consumo de CPU...
[CPU Intensive] PID: 2964 - Iniciando consumo de CPU...
[CPU Intensive] PID: 2965 - Iniciando consumo de CPU...
usuario@usuario-VirtualBox:~$
```

Ejecuta 5 programas utilizando CPU

```
usuario@usuario-VirtualBox: ~  
usuario@usuario-VirtualBox:~$ sar -u 1 10  
Linux 6.11.0-25-generic (usuario-VirtualBox) 17/06/25 _x86_64_ (4 CPU)  
  
13:29:08 CPU %user %nice %system %iowait %steal %idle  
13:29:09 all 2,74 0,00 0,25 0,00 0,00 97,01  
13:29:10 all 4,02 0,00 1,51 0,00 0,00 94,47  
13:29:11 all 2,53 0,00 0,76 0,00 0,00 96,71  
13:29:12 all 3,82 0,00 3,56 0,00 0,00 92,62  
13:29:13 all 2,50 0,00 0,75 0,00 0,00 96,75  
13:29:14 all 2,65 0,00 0,27 0,00 0,00 97,08  
13:29:15 all 2,70 0,00 1,47 0,25 0,00 95,58  
13:29:16 all 2,27 0,00 0,00 0,00 0,00 97,73  
13:29:17 all 2,50 0,00 1,25 0,00 0,00 96,25  
13:29:18 all 2,51 0,00 1,00 0,00 0,00 96,49  
Media: all 2,82 0,00 1,08 0,03 0,00 96,07  
usuario@usuario-VirtualBox:~$
```

SAR para tener estadísticas detalladas.

```
usuario@usuario-VirtualBox: ~  
usuario@usuario-VirtualBox:~$ ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu | head  
PID PPID CMD %MEM %CPU  
2992 2985 htop 0.0 2.2  
1416 1144 /usr/bin/gnome-shell 4.9 2.0  
2962 2954 python3 cpu_intensivo_ubuntu 0.1 1.5  
2964 2954 python3 cpu_intensivo_ubuntu 0.1 1.5  
2965 2954 python3 cpu_intensivo_ubuntu 0.1 1.5  
2961 2954 python3 cpu_intensivo_ubuntu 0.1 1.5  
2963 2954 python3 cpu_intensivo_ubuntu 0.1 1.5  
2750 1144 /usr/libexec/gnome-terminal 0.8 0.5  
3013 3006 top 0.0 0.2  
usuario@usuario-VirtualBox:~$
```

Identifica la jerarquía de procesos


```
TOP ver la distribución del CPU
top - 13:30:27 up 1:38, 1 user, load average: 0,38, 0,31, 0,16
Tareas: 231 total, 1 ejecutar, 230 hibernar, 0 detener, 0 zombie
%Cpu(s): 2,7 us, 1,3 sy, 0,0 ni, 95,7 id, 0,1 wa, 0,0 hi, 0,2 si, 0,0 st
MiB Mem : 7940,9 total, 6120,8 libre, 1169,7 usado, 933,7 búf/caché
MiB Intercambio: 4096,0 total, 4096,0 libre, 0,0 usado, 6771,2 dispon
```

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
1416	usuario	20	0	4892588	396372	148320	S	7,0	4,9	2:03.42	gnome-s+
2992	usuario	20	0	19936	4940	3660	S	2,6	0,1	0:07.27	htop
2750	usuario	20	0	581612	69848	50100	S	1,7	0,9	0:12.39	gnome-t+
2964	usuario	20	0	29120	9356	6540	S	1,7	0,1	0:05.28	python3
2961	usuario	20	0	29120	9388	6572	S	1,3	0,1	0:05.33	python3
2962	usuario	20	0	29120	9356	6540	S	1,3	0,1	0:05.46	python3
2963	usuario	20	0	29120	9360	6544	S	1,3	0,1	0:05.32	python3
2965	usuario	20	0	29120	9340	6524	S	1,0	0,1	0:05.43	python3
2899	root	20	0	0	0	0	I	0,7	0,0	0:01.26	kworker+
482	systemd+	20	0	17556	7500	6732	S	0,3	0,1	0:02.29	systemd+
1	root	20	0	23164	13932	9452	S	0,0	0,2	0:01.19	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.00	pool_wo+
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+
5	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+
7	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker+

#Creación de Deadlock

#Objetivo:

- Investigar como se genera un deadlock y como el Sistema Operativo responde.

#Métodos:

- Se crearon dos scripts que compiten por dos recursos (archivos).

```
import threading
import time
import os

lock1 = threading.Lock()
lock2 = threading.Lock()

def hilo1():
    with lock1:
        print(f"[Hilo1] PID: {os.getpid()} - Tengo lock1, esperando lock2...")
        time.sleep(3)
        with lock2:
            print("[Hilo1] Nunca llegará aquí") # Deadlock evita es
to

def hilo2():
    with lock2:
        print(f"[Hilo2] PID: {os.getpid()} - Tengo lock2, esperando lock1...")
        time.sleep(3)
```

```

        with lock1:
            print("[Hilo2] Nunca llegará aquí") # Deadlock evita es
to

print(f"[Main] PID: {os.getpid()} - Busca este proceso en htop/ps")
t1 = threading.Thread(target=hilo1)
t2 = threading.Thread(target=hilo2)
t1.start(); t2.start()
t1.join(); t2.join() # ;Paréntesis añadidos!

```


- Se intento generar un deadlock haciendo que los procesos esperen recursos mutuamente exclusivos.

#Resultados:

- Capturas de pantalla mostrando los errores de deadlock.
- Descripción de como el Sistema Operativo responde al deadlock.

#Análisis:

- Propuesta para evitar o resolver deadlocks en sistemas reales.



```

usuario@usuario-VirtualBox: ~
usuario@usuario-VirtualBox:~$ nano deadlock_ubuntu.py
usuario@usuario-VirtualBox:~$ python3 deadlock_ubuntu.py
[Main] PID: 3767 - Busca este proceso en htop
[Hilo1] PID: 3767 - Tengo lock1, esperando lock2...
[Hilo2] PID: 3767 - Tengo lock2, esperando lock1...

```

Deadlock simple:

Los hilos se quedan "colgados" sin completar su ejecución.

El programa **no termina** (a menos que lo fuerces con Ctrl+C).

PROCESOS

3945	usuario	20	0	19692	5092	3812	S	0.0	0.1	0:00.00	bash
3953	usuario	20	0	172M	10304	6720	S	0.0	0.1	0:00.01	python3 deadl
3954	usuario	20	0	172M	10304	6720	S	0.0	0.1	0:00.00	python3 deadl
3955	usuario	20	0	172M	10304	6720	S	0.0	0.1	0:00.00	python3 deadl

```

usuario@usuario-VirtualBox: ~
0[| 0.6%] Tasks: 122, 357 thr, 110 kthr; 2 runni
1[|| 1.9%] Load average: 0.13 0.25 0.25
2[| 1.2%] Uptime: 04:34:30
3[|| 3.1%]
Mem[|||||] 967M/7.75G
Swp[ ] 0K/4.00G

Main I/O
Sort by
PID 3922 usuario 20 0 456M 28580 21284 S 0.0 0.4 0:00.0
USER 3923 usuario 20 0 456M 28580 21284 S 0.0 0.4 0:00.0
PRIORITY 3924 usuario 20 0 19692 5060 3780 S 0.0 0.1 0:00.0
NICE 3936 usuario 20 0 40560 19248 10928 S 0.0 0.2 0:00.0
M_VIRT 3939 usuario 20 0 456M 28456 21288 S 0.0 0.3 0:00.0
M_RESIDENT 3940 usuario 20 0 456M 28456 21288 S 0.0 0.3 0:00.0
M_SHARE 3941 usuario 20 0 456M 28456 21288 S 0.0 0.3 0:00.0
STATE 3943 usuario 20 0 456M 28456 21288 S 0.0 0.3 0:00.0
PERCENT_CPU 3944 usuario 20 0 456M 28456 21288 S 0.0 0.3 0:00.0
PERCENT_MEM 3945 usuario 20 0 19692 5092 3812 S 0.0 0.1 0:00.0
TIME 3953 usuario 20 0 172M 10304 6720 S 0.0 0.1 0:00.0
Command 3954 usuario 20 0 172M 10304 6720 S 0.0 0.1 0:00.0
3955 usuario 20 0 172M 10304 6720 S 0.0 0.1 0:00.0
Enter Sort Esc Cancel

```

Agrupado por estados

```

usuario@usuario-VirtualBox: ~
usuario@usuario-VirtualBox:~$ ps aux | grep python3
root      1111  0.0  0.2 120904 22984 ?        Ssl  11:51   0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unatt
ended-upgrade-shutdown --wait-for-signal
usuario   2945  0.0  0.2 40560 18980 ?        S   13:23   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   2961  2.9  0.1 29120 9388 pts/0    S   13:24   5:24 python3 cpu_intensivo_ubuntu.py
usuario   2962  2.9  0.1 29120 9356 pts/0    S   13:24   5:30 python3 cpu_intensivo_ubuntu.py
usuario   2963  2.9  0.1 29120 9360 pts/0    S   13:24   5:24 python3 cpu_intensivo_ubuntu.py
usuario   2964  2.9  0.1 29120 9356 pts/0    S   13:24   5:27 python3 cpu_intensivo_ubuntu.py
usuario   2965  2.9  0.1 29120 9340 pts/0    S   13:24   5:23 python3 cpu_intensivo_ubuntu.py
usuario   2997  0.0  0.2 40560 19076 ?        S   13:27   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   3915  0.0  0.2 40560 19176 ?        S   16:23   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   3936  0.0  0.2 40560 19248 ?        S   16:23   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   3953  0.0  0.1 176688 10304 pts/1    Sl+  16:23   0:00 python3 deadlock_ubuntu.py
usuario   3964  0.1  0.2 40560 19144 ?        S   16:28   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   3981  0.0  0.0 17832 2384 pts/4     S+   16:28   0:00 grep --color=auto python3
usuario@usuario-VirtualBox:~$

```

Todos los procesos de python3

```

usuario@usuario-VirtualBox:~$ pkill -f "python3 cpu_intensivo_ubuntu.py"
usuario@usuario-VirtualBox:~$ ps aux | grep python3
root      1111  0.0  0.2 120904 22984 ?        Ssl  11:51   0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unatt
ended-upgrade-shutdown --wait-for-signal
usuario   2945  0.0  0.2 40560 18980 ?        S   13:23   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   2997  0.0  0.2 40560 19076 ?        S   13:27   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   3915  0.0  0.2 40560 19176 ?        S   16:23   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   3936  0.0  0.2 40560 19248 ?        S   16:23   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   3953  0.0  0.1 176688 10304 pts/1    Sl+  16:23   0:00 python3 deadlock_ubuntu.py
usuario   3964  0.0  0.2 40560 19144 ?        S   16:28   0:00 /usr/bin/python3 /usr/bin/gnome-terminal --wait
usuario   3996  0.0  0.0 17832 2384 pts/4     S+   16:32   0:00 grep --color=auto python3
usuario@usuario-VirtualBox:~$

```

Finalización de uno de los procesos

#CONCLUSION:

El laboratorio permitió comprender como los sistemas operativos gestionan proceso, optimizan recursos y manejan conflictos, aplicando tanto teoría (algoritmos de scheduling) como practica (herramientas de monitoreo y depuración).