UNIVERSIDAD DE LA INTEGRACIÓN DE LAS AMÉRICAS

FACULTAD DE INGENIERÍA.

MATERIA

Sistemas Operativos.

TÍTULO

LABORATORIO DE ANÁLISIS DE SISTEMAS OPERATIVOS

Mg. Alan Vladimir Dioses Echegaray.

Lucio Vera.

Estudiante

Priscila Jazmín Rivas Gaona

Asunción- Paraguay.

2025.

Informe de Laboratorio 1 – Gestión de Procesos

Estados de Procesos

Se creó un script en Python que simula el ciclo de vida de un proceso pasando por los estados: Nuevo, Listo, Ejecutando, Bloqueado y Terminado. Para representar estos estados, se utilizaron las funciones time.sleep() y print(), mostrando los cambios de estado de manera secuencial.

Los estados fueron identificados mediante mensajes impresos por el programa. Cada estado tuvo una pausa breve para poder observarlo, y se tomaron capturas de pantalla en cada uno.

Los estados registrados son:

- Nuevo
- Listo
- Ejecutando
- Bloqueado
- Terminado

Medición de tiempos de transición

Se midieron los tiempos de transición entre cada estado utilizando el módulo time de Python. Las funciones time.time() fueron utilizadas para calcular los segundos exactos entre los cambios de estado.

Los datos obtenidos fueron exportados al archivo datos_mediciones.xlsx para su análisis.

Este archivo contiene los tiempos exactos entre:

- Nuevo → Listo
- Listo → Ejecutando

- Ejecutando → Bloqueado

- Bloqueado → Terminado

Scheduling de un Sistema Operativo

Para analizar el comportamiento del planificador del sistema operativo, se abrieron múltiples terminales en Ubuntu y se ejecutaron simultáneamente scripts que realizaban cálculos intensivos.

Durante esta actividad se utilizó la herramienta htop para observar cómo el sistema operativo distribuye el uso del CPU entre los diferentes procesos.

Observaciones:

Los procesos fueron ejecutados de manera alternada por el CPU.

Se pudo observar el cambio dinámico de prioridad y la cantidad de CPU asignado a cada proceso.

Comparación con algoritmos de planificación:

FIFO (First In, First Out): los procesos ejecutan en el orden en que llegan, uno tras otro.

Round Robin: se asigna un tiempo igual a cada proceso de forma circular. Este fue el comportamiento más cercano al observado en Ubuntu.

Creación de Deadlock

Se programó una simulación de interbloqueo utilizando threading.Lock() en Python. Dos hilos intentaban adquirir recursos en distinto orden, lo que provocó una espera circular.

Durante la ejecución, los procesos quedaron bloqueados sin poder avanzar, lo que representa un deadlock.

Análisis:

Los procesos no pudieron continuar porque cada uno retenía un recurso que el otro necesitaba.

El sistema operativo no interrumpió automáticamente este estado.

Se comprobó en htop que los procesos seguían activos pero sin consumir CPU.