INTEGRANTES

- David Arias Rueda
- Juan David Carvajal Lozano
- Paula Andrea Gómez Aldana

PARCIAL 2
SISTEMAS OPERATIVOS

Planificación por Prioridades Dinámicas:
Implementar un algoritmo de planificación que permita a los procesos cambiar dinámicamente sus prioridades en función de su comportamiento y estado. La implementación puede debe ser en Python y en C.

Ejemplo, se podría usar el algoritmo de planificación de retroalimentación multinivel (MLFQ) donde los procesos se mueven entre colas con diferentes prioridades en función de su tiempo de CPU utilizado.

Recomendación:
Para implementar un algoritmo de planificación con prioridades dinámicas, se necesita una forma de calcular y ajustar las prioridades de los procesos en función de su comportamiento. Por ejemplo, se podría aumentar la prioridad de un proceso si ha esperado mucho tiempo en la cola de listos, o reducir su prioridad si ha utilizado una gran cantidad de tiempo de CPU recientemente. (Se adjunta modelo).

SOLUCIÓN

Programa en C

Este programa se ejecutó 15 veces con el fin de tomar su tiempo de ejecución y hacer un análisis más específico, los siguientes son los tiempos de ejecución:

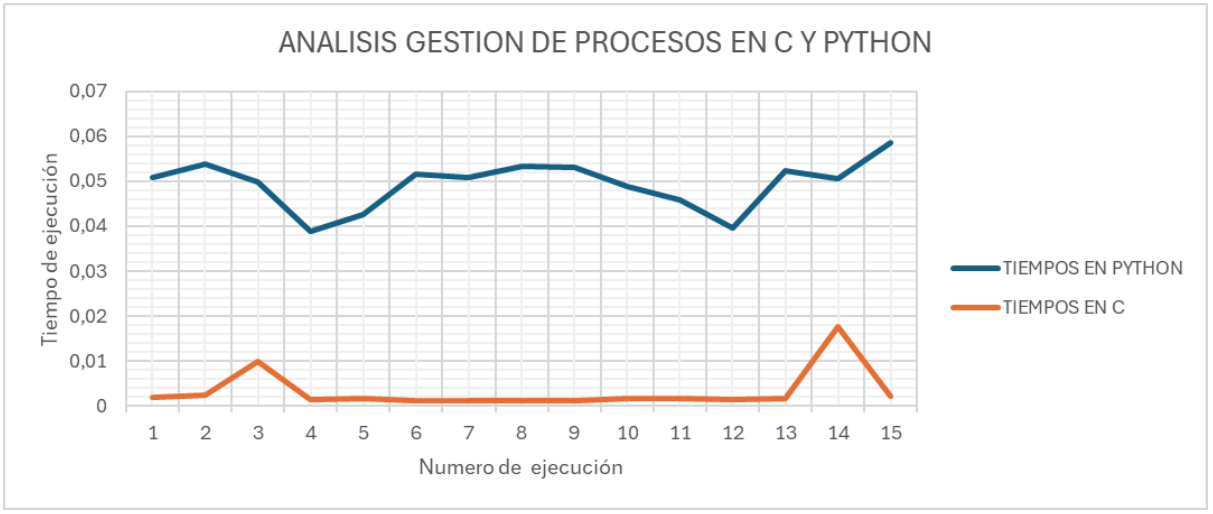
```
Tiempo de ejecución 1: 0.001832 segundos
Tiempo de ejecución 2: 0.002403 segundos
Tiempo de ejecución 3: 0.009878 segundos
Tiempo de ejecución 4: 0.001431 segundos
Tiempo de ejecución 5: 0.001742 segundos
Tiempo de ejecución 6: 0.001135 segundos
Tiempo de ejecución 7: 0.001088 segundos
Tiempo de ejecución 8: 0.001040 segundos
Tiempo de ejecución 9: 0.001146 segundos
Tiempo de ejecución 10: 0.001696 segundos
Tiempo de ejecución 11: 0.001572 segundos
Tiempo de ejecución 12: 0.001489 segundos
Tiempo de ejecución 13: 0.001741 segundos
Tiempo de ejecución 14: 0.017694 segundos
Tiempo de ejecución 15: 0.002042 segundos
```

Programa en Python

Este programa se ejecutó 15 veces con el fin de tomar su tiempo de ejecución y hacer un análisis más específico, los siguientes son los tiempos de ejecución:

```
Tiempo de ejecución 1: 0.050743 segundos
Tiempo de ejecución 2: 0.053845 segundos
Tiempo de ejecución 3: 0.049945 segundos
Tiempo de ejecución 4: 0.038767 segundos
Tiempo de ejecución 5: 0.042577 segundos
Tiempo de ejecución 6: 0.051465 segundos
Tiempo de ejecución 7: 0.050832 segundos
Tiempo de ejecución 8: 0.053225 segundos
Tiempo de ejecución 9: 0.052999 segundos
Tiempo de ejecución 10: 0.048735 segundos
Tiempo de ejecución 11: 0.045930 segundos
Tiempo de ejecución 12: 0.039680 segundos
Tiempo de ejecución 13: 0.052324 segundos
Tiempo de ejecución 14: 0.050699 segundos
Tiempo de ejecución 15: 0.058494 segundos
```

Análisis entre tiempos de ejecución en la gestión de procesos, según el lenguaje de programación (C y Python)



Teniendo en cuenta la gráfica anterior se evidencia que los tiempos de ejecución presentados en Python son más altos que los presentados en C, esto puede deberse a que Python es un lenguaje de programación interpretado, a diferencia de C que es un lenguaje de programación compilado, lo que afecta directamente a su tiempo de ejecución.