UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingeniería

Sistemas Operativos

Tarea 2. Comparación de Planificadores

Profesor: Dr. Gunnar Eyal Wolf Iszaevich

Grupo: 8

Alumno:

- Alvarez Salgado Eduardo Antonio (321335630)

Fecha de entrega: 28 de octubre de 2025

Semestre: 2026-1

Introducción

A lo largo de este proyecto se buscará desarrollar un código en lenguaje de programación Python el cual permita simular y comparar diferentes algoritmos de planificación de procesos en un sistema operativo, evaluando su comportamiento con cargas de trabajo generadas aleatoriamente. Para ello se implementan planificadores clásicos como First Come, First Served (FCFS), Round Robin con quantum 1 (RR1), Round Robin con quantum 4 (RR4), Shortest Process Next (SPN) y un esquema de colas múltiples con retroalimentación (MFQ).

Esto con la finalidad de analizar cómo se comporta cada algoritmo frente a procesos con tiempos de llegada y ráfagas distintos, y representar los resultados en forma de métricas (tiempo de retorno, espera y penalización) así como en un esquema visual de ejecución que muestre la utilización del procesador en cada instante de tiempo.

Desarrollo

1.- Cargas realizadas

La función crear_procesos construye la carga de trabajo, en donde mediante un bucle se genera una secuencia de tuplas las cuales serán probadas en todos los planificadores. Estas se hacen de manera aleatoria y son identificadas mediante las letras del abecedario (A, B, C, D, E).

Los tiempos de llegada se generan con intervalos aleatorios entre 0 y un valor máximo (max_gap), lo cual permite que existan huecos de inactividad en el procesador. Por otro lado, las ráfagas de CPU se asignan de forma aleatoria entre 1 y un valor máximo (max_burst).

2.- Cálculo de las métricas

Esto se lleva a cabo mediante la función calcular_metricas, en donde se determinan:

- **T (Tiempo de retorno)**: tiempo total de permanencia del proceso en el sistema.
- **E (Espera):** tiempo que el proceso pasa en cola sin ejecutarse.
- **P (Penalización):** relación entre el tiempo de retorno y la ráfaga.

Estas métricas se imprimen para cada uno de los planificadores mediante la función imprimir_metricas.

3.- Planificadores

Para cada uno de los planificadores, se crea una función que sigue su lógica correspondiente.

- **First Come, First Served (FCFS):** Atiende los procesos en orden de llegada en donde el proceso anterior debe ejecutarse por completo antes de pasar al siguiente. Si un proceso aún no llega, se insertan - en la tira para reflejar inactividad.
- **Round Robin (RR1 y RR4):** Cada proceso recibe un número de hasta quantum ticks, en donde si no termina, se reencola para permitir alternar entre procesos.
- **Shortest Process Next (SPN):** Este planificador selecciona siempre el proceso más corto entre los disponibles, en donde si no hay procesos disponibles, se generan huecos representados con "-".

4.- Algoritmo de colas múltiples

Para este punto se desarrolló el algoritmo Multilevel Feedback Queue (MFQ), el cual implementa un esquema de colas múltiples con retroalimentación.

En este algoritmo se definen varias colas, cada una con un quantum distinto (en este caso [1,2,4]). Los procesos son generados y entran al nivel más alto, en donde si no finalizan con el número de quantums, bajan al siguiente nivel. Esto permite dar prioridad a la finalización de procesos más cortos y dejar hasta el final los más largos.

5.- Esquema visual

La ejecución de procesos se representa como una tira de caracteres, identificados con su respectiva letra. Cada carácter corresponde a un tick de reloj y cuando existan huecos en los planificadores, son representados mediante un "-", permitiendo así visualizar la manera en que cada planificador maneja los procesos y en qué momentos se generan huecos de inactividad.

Resultados obtenidos:

```
- Ronda 1:
  A: 0, t=4; B: 0, t=3; C: 0, t=1; D: 3, t=2; E: 5, t=1 (tot:11)
FCFS: T=6.40, E=4.20, P=4.17
AAAABBBCDDE
RR1: T=6.60, E=4.40, P=3.05
ABCABDAEBDA
RR4: T=6.40, E=4.20, P=4.17
AAAABBBCDDE
SPN: T=4.20, E=2.00, P=1.72
CBBBDDEAAAA
MFQ: T=6.40, E=4.20, P=2.85
ABCDAAEBBDA

- Ronda 2:
  A: 0, t=4; B: 4, t=4; C: 5, t=3; D: 9, t=4; E: 9, t=4 (tot:19)
FCFS: T=6.00, E=2.20, P=1.60
AAAABBBBCCCDDDDEEEE
RR1: T=7.40, E=3.60, P=1.93
AAAABCBCBCDEBDEDEDE
RR4: T=6.00, E=2.20, P=1.60
AAAABBBBCCCDDDDEEEE
SPN: T=6.00, E=2.20, P=1.60
AAAABBBBCCCDDDDEEEE
MFQ: T=8.20, E=4.40, P=2.13
AAAABCBBCCDEDDEEBDE

- Ronda 3:
  A: 0, t=5; B: 2, t=3; C: 8, t=1; D: 11, t=1; E: 13, t=2 (tot:12)
FCFS: T=3.00, E=0.60, P=1.20
AAAAABBBC--D-EE
RR1: T=3.40, E=1.00, P=1.25
AABABABAC--D-EE
RR4: T=3.40, E=1.00, P=1.25
AAAABBBAC--D-EE
SPN: T=3.00, E=0.60, P=1.20
AAAAABBBC--D-EE
MFQ: T=3.20, E=0.80, P=1.19
AAABBBAAC--D-EE
```

```
- Ronda 4:
  A: 0, t=4; B: 0, t=5; C: 2, t=2; D: 5, t=3; E: 11, t=2 (tot:16)
FCFS: T=7.20, E=4.00, P=2.56
AAAABBBBBCCDDDEE
RR1: T=8.80, E=5.60, P=2.70
ABACBACDBADBEDBE
RR4: T=7.80, E=4.60, P=2.59
AAAABBBBCCDDDBEE
SPN: T=6.20, E=3.00, P=1.93
AAAACCDDDBBBBBEE
MFQ: T=9.00, E=5.80, P=2.64
ABCAADBBCDDEEABB

- Ronda 5:
  A: 0, t=2; B: 0, t=2; C: 3, t=5; D: 7, t=4; E: 10, t=5 (tot:18)
FCFS: T=5.20, E=1.60, P=1.46
AABBCCCCCDDDDEEEEE
RR1: T=6.20, E=2.60, P=1.74
ABABCCCDCDCEDEDEEE
RR4: T=5.80, E=2.20, P=1.57
AABBCCCCDDDDCEEEEE
SPN: T=5.20, E=1.60, P=1.46
AABBCCCCCDDDDEEEEE
MFQ: T=7.40, E=3.80, P=2.05
ABACBCCDDDEEECCDEE
PS C:\Users\eanto\OneDrive\Escritorio\PruebasT2>
```

Conclusiones

El código desarrollado logra simular y comparar distintos algoritmos de planificación de procesos bajo condiciones de llegadas y ráfagas aleatorias.

La simulación no solo ofrece los valores numéricos de métricas de rendimiento, sino que también genera un esquema visual que facilita la comprensión de la ejecución y los huecos de inactividad. De esta forma, se puede observar de manera práctica cómo varía la eficiencia y el tiempo de respuesta según el planificador empleado.