Nombre: Dennys Alexander Pucha Carrera

Paralelo: 4to “A” **Fecha:** 09/07/2023

Asignatura: Sistemas Operativos

Docente: Ing. Hernán Leonardo Torres Carrión M.Sc.

ENSAYO Nº 8

1. **Tema**

   Algoritmos de Planificación de la CPU

2. **Antecedentes**

   En este ensayo, se explorará los algoritmos de planificación de procesos o de la CPU, con el objetivo de comprender su funcionamiento y analizar su impacto en el rendimiento del sistema. Se abordará seis algoritmos más utilizados en la industria: FIFO, SJF, Prioridades, RR, Múltiples niveles y Múltiples niveles retroalimentados.

   Durante este estudio, se enfocará el ensayo en examinar detenidamente las características distintivas de cada algoritmo. Se comprenderá cómo se diferencian en términos de su enfoque de planificación, la forma en que asignan los recursos de la CPU y cómo manejan la prioridad de los procesos.

   Además, para brindar una comprensión más clara de cada algoritmo, se presentará ejemplos prácticos. Estos ejemplos permitirán visualizar cómo se lleva a cabo la planificación de procesos en diferentes situaciones y cómo afecta el tiempo de ejecución y la espera de los procesos.

   A lo largo de este ensayo, podrás descubrir las ventajas y desventajas de cada algoritmo y cómo pueden influir en el rendimiento general del sistema operativo. Al comprender las características de estos algoritmos de planificación de procesos, los estudiantes de computación como yo podrán apreciar la importancia de seleccionar el algoritmo adecuado para optimizar la eficiencia y la capacidad de respuesta del sistema.

3. **Descripción**

   Antes de adentrarse como tal en los algoritmos de planificación de la CPU es de mucha importancia de que se trata esto de la planificación de la CPU a breves rasgos:

   La planificación de la CPU es un proceso para determinar qué proceso será propietario de la CPU para su ejecución mientras otro proceso está en espera. La tarea principal de la programación de la CPU es asegurarse de que siempre que la CPU permanezca inactiva, el sistema operativo seleccione al menos uno de los procesos disponibles en la cola lista para su ejecución. [1]

   En base a esta idea se puede decir que la planificación de la CPU es un componente esencial en los sistemas operativos para administrar eficientemente el tiempo de ejecución de los procesos en una computadora. Su objetivo principal es asegurarse de que la CPU esté ocupada en todo momento y que los procesos se ejecuten de manera justa y eficiente.

Para la planificación de tiempo de la CPU existen numerosos algoritmos que se utilizan en los sistemas operativos para determinar el orden en el que los procesos serán asignados a la CPU. Cada algoritmo tiene sus propias características y objetivos, y la elección del algoritmo adecuado depende del entorno de ejecución y de los requisitos del sistema.

Como objeto de estudio se han seleccionado los 6 algoritmos más comunes y más utilizados (FIFO, SJF, Prioridades, RR, Múltiples niveles y Múltiples niveles retroalimentados) hoy en día, a continuación, se muestra una tabla comparativa para cada uno de estos algoritmos: Es importante ver cómo es que funcionan cada uno de estos algoritmos mediante ejemplos para explicar de mejor manera se va a hacer con los siguientes procesos:

Algoritmo	**Características**	**Ventajas**	**Desventajas**
FIFO (Primero en llegar)	- Basado en el orden de llegada	- Fácil de implementar	- Tiempo de espera alto
SJF (Trabajo más corto)	- Selecciona el proceso más corto primero	- Tiempo de espera promedio bajo	- Dificultad para predecir la duración de los procesos
Programación de prioridades	- Asigna la CPU según la prioridad del proceso	- Permite priorizar procesos importantes	- Posibilidad de inanición del proceso [2]
Round Robin	- Asigna a cada proceso un intervalo de tiempo fijo	- Uso equitativo de la CPU	- Alta latencia para procesos largos
Programación de múltiples niveles	- Divide los procesos en diferentes colas con necesidades de programación específicas	- Uso eficiente de los recursos del sistema	- Problema de inanición, inflexibilidad
Programación de múltiples niveles retroalimentados	- Permite que los procesos se muevan entre colas y ajusta dinámicamente las prioridades	- Mayor flexibilidad y adaptabilidad	- Gastos generales de CPU, complejidad [3]

EJEMPLO:

- Se tiene la siguiente cola de procesos simular su ejecución mediante los distintos algoritmos de planificación de la CPU:

1.FIFO: Basado en el orden de llegada (Primero en entrar es el Primero en Salir)

Proceso A: Tiempo de llegada = 0 ms, Duración = 3 ms
Proceso B: Tiempo de llegada = 1 ms, Duración = 4 ms
Proceso C: Tiempo de llegada = 2 ms, Duración = 2 ms

Tiempo (ms)	**Proceso en ejecución**	**Procesos en espera**
0	-	A
1	A	B
2	A	B, C
3	A	B, C

4	B	C
5	B	C
6	B	C
7	B	C
8	C	-
9	C	-
10	-	-

En este ejemplo, el proceso A llega al tiempo 0 ms y comienza su ejecución. Luego, el proceso B llega al tiempo 1 ms y se coloca en espera ya que el proceso A aún no ha finalizado. El proceso A continúa ejecutándose hasta que finaliza su duración total al tiempo 3 ms.

En el tiempo 3 ms, el proceso B pasa a ejecutarse ya que es el siguiente en la cola de espera. El proceso B tiene una duración de 4 ms, por lo que se ejecuta rápidamente y finaliza al tiempo 7 ms.

Después de que el proceso B finaliza su ejecución, el proceso C toma su lugar y comienza a ejecutarse en el tiempo 2 ms. El proceso C tiene una duración de 2 ms y continúa ejecutándose hasta que finaliza su duración total.

2. SJF (Trabajo más corto): Selecciona el proceso más corto primero
Proceso A: Tiempo de llegada = 0 ms, Duración = 3 ms
Proceso B: Tiempo de llegada = 1 ms, Duración = 4 ms
Proceso C: Tiempo de llegada = 2 ms, Duración = 2 ms

Tiempo (ms)	**Proceso en ejecución**	**Procesos en espera**
0	-	A
1	A	B
2	A	B, C
3	A	B, C
4	C	B
5	C	B
6	B	-
7	B	-
8	B	-
9	B	-
10	-	-

Como se ve el proceso A se ejecuta primero ya que llega primero pero luego ya que hay dos procesos en espera selecciona el proceso más corto y lo ejecuta en este caso el C y finaliza la ejecución con el proceso B

3. Programación de prioridades: Asigna la CPU según la prioridad del proceso
Proceso A: Tiempo de llegada = 0 ms, Duración = 3 ms PRIORIDAD: 3
Proceso B: Tiempo de llegada = 0 ms, Duración = 4 ms PRIORIDAD: 2
Proceso C: Tiempo de llegada = 0 ms, Duración = 2 ms PRIORIDAD: 1

Tiempo (ms)	Proceso en ejecución	Procesos en espera
0	-	B, A
1	C	B, A
2	C	B, A
3	B	A
4	B	A
5	B	A
6	B	A
7	A	-
8	A	-
9	A	-
10	-	-

Como su nombre lo dice es por prioridades aquí se asigna directamente la prioridad, pero en el algoritmo se selecciona en base a parámetros o características según sea el proceso.

4. Round Robin: Asigna a cada proceso un intervalo de tiempo fijo
Proceso A: Tiempo de llegada = 0 ms, Duración = 3 ms
Proceso B: Tiempo de llegada = 1 ms, Duración = 4 ms
Proceso C: Tiempo de llegada = 2 ms, Duración = 2 ms
Tiempo fijo: 3 ms

Tiempo (ms)	Proceso en ejecución	Procesos en espera
0	-	A
1	A	
2	A	
3	A	B, C
4	B	
5	B	
6	B	C
7	C	B
8	C	B
9	-	B
10	B	-
11	-	-

Al asignarse un tiempo fijo el proceso solo dispone de tal tiempo para ejecutarse en caso de no alcanzar su ejecución debe esperar hasta que se le asigne de nuevo este tiempo fijo para terminar su ejecución, en este ejemplo el proceso B no alcanzo a ejecutarse en 3 ms entonces debe esperar hasta que se le asigne de nuevo tiempo en la CPU tal y como se muestra en la tabla.

5. Programación de múltiples niveles: Divide los procesos en diferentes colas con necesidades de programación específicas
Proceso A: Tiempo de llegada = 0 ms, Duración = 3 ms PRIORIDAD: 3
Proceso B: Tiempo de llegada = 0 ms, Duración = 4 ms PRIORIDAD: 2
Proceso C: Tiempo de llegada = 0 ms, Duración = 2 ms PRIORIDAD: 1

Tiempo (ms)	Cola 1 (Alta prioridad)	Cola 2 (Media prioridad)	Cola 3 (Baja prioridad)
0	C (2 ms)	B (4 ms)	A (3 ms)
1	C (1 ms)	B (4 ms)	A (3 ms)
2	C (1 ms)	B (3 ms)	A (3 ms)
3	C (1 ms)	B (2 ms)	A (3 ms)
4	C (1 ms)	B (1 ms)	A (3 ms)
5	C (1 ms)	A (2 ms)	A (3 ms)
6	C (1 ms)	A (1 ms)	A (3 ms)
7	C (1 ms)	A (1 ms)	A (2 ms)
8	C (1 ms)	A (1 ms)	A (1 ms)
9	C (1 ms)	A (1 ms)	-
10	-	A (1 ms)	-
11	-	-	-

En este ejemplo, se dividen los procesos en tres colas con diferentes prioridades. Los procesos se ejecutan en orden de prioridad, comenzando por la Cola 1 (Alta prioridad), luego la Cola 2 (Media prioridad) y finalmente la Cola 3 (Baja prioridad). Si hay múltiples procesos en la misma cola, se utiliza el algoritmo de Primero en Llegar, Primero en Servir (FCFS) para seleccionar el siguiente proceso a ejecutar.

En el tiempo 0 ms, los procesos A, B y C llegan al sistema y se colocan en sus respectivas colas según su prioridad. El proceso C, con la prioridad más alta, comienza su ejecución. Luego, el proceso B se ejecuta en la siguiente iteración, seguido del proceso A. Esto se repite hasta que todos los procesos han completado su duración.

6. Programación de múltiples niveles retroalimentados: Permite que los procesos se muevan entre colas y ajusta dinámicamente las prioridades

Proceso A: Tiempo de llegada = 0 ms, Duración = 3 ms
Proceso B: Tiempo de llegada = 1 ms, Duración = 4 ms
Proceso C: Tiempo de llegada = 2 ms, Duración = 2 ms

Tiempo (ms)	Cola 1 (Alta prioridad)	Cola 2 (Media prioridad)	Cola 3 (Baja prioridad)
0	A (3 ms)	-	-
1	A (2 ms)	B (4 ms)	-
2	A (2 ms)	B (3 ms)	C (2 ms)
3	A (1 ms)	B (3 ms)	C (2 ms)
4	A (1 ms)	B (2 ms)	C (2 ms)
5	A (1 ms)	B (1 ms)	C (2 ms)
6	A (1 ms)	B (1 ms)	C (1 ms)
7	A (1 ms)	B (1 ms)	C (1 ms)
8	A (1 ms)	B (1 ms)	C (1 ms)
9	-	B (1 ms)	C (1 ms)
10	-	-	C (1 ms)
11	-	-	C (1 ms)
12	-	-	C (1 ms)
13	-	-	-

En este ejemplo, se utilizan tres colas con diferentes prioridades. Los procesos se colocan inicialmente en la Cola 1 (Alta prioridad) y luego se mueven a colas de menor prioridad si no se han completado dentro de un cierto límite de tiempo. Cada cola tiene un límite de tiempo asignado antes de que un proceso sea movido a la siguiente cola de prioridad inferior.

En el tiempo 0 ms, el proceso A llega al sistema y se coloca en la Cola 1 (Alta prioridad). En el tiempo 1 ms, el proceso B llega al sistema y se coloca en la misma cola. En el tiempo 2 ms, el proceso C llega al sistema y se coloca en la Cola 2 (Media prioridad).

Cada proceso se ejecuta en su cola asignada durante un cierto período de tiempo antes de pasar al siguiente proceso en la cola. Si un proceso no se ha completado dentro de ese límite de tiempo, se mueve a la cola de prioridad inferior. Esto permite que los procesos más cortos se completen más rápidamente y se les dé prioridad en la ejecución.[4]

4. Conclusiones

- Cada algoritmo de planificación tiene sus ventajas y desventajas, es así que no se puede decir que uno es mejor que otro ya que dependerá de las necesidades que se tengan sobre los procesos para que estos sean ejecutados.
- Muchos de estos algoritmos tienen similitudes en cuanto a la forma en que seleccionan que algoritmo tiene una mayor o menor prioridad sin embargo varían en los parámetros o características para seleccionar la prioridad.
- La importancia de los algoritmos de planificación de la CPU es muy alta ya que permite definir o seleccionar que proceso será el siguiente a ejecutar lo que permite a la CPU trabajar de manera más ordenada y eficiente.

5. Bibliografía

[1] G. 99, “CPU Scheduling Algorithms,” Guru99, 2021. [Online]. Available: https://www.guru99.com/cpu-scheduling-algorithms.html. [Accessed: 9-Jul-2023].

[2] G. for Geeks, “CPU Scheduling in Operating Systems,” GeeksforGeeks, 2021. [Online]. Available: https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/. [Accessed: 9-Jul-2023].

[3] W. Plus Valencia, “Algoritmos de planificación FCFS, SJF, SRTF, Round Robin,” Web Plus Valencia, 2019. [Online]. Available: https://webplusvalencia.es/algoritmos-de-planificacion-fcfs-sjf-srtf-round-robind/. [Accessed: 9-Jul-2023].

[4] W. Stallings, Sistemas operativos: aspectos internos y principios de diseño, Pearson Educación, 2009. [Online]. Disponible en: https://books.google.com/books/about/Sistemas_operativos.html?id=0fYjAQAAMAAJ. [Accedido: 9-Jul-2023].