# Proyecto Primer Parcial

# "Simulación y Comparación de Algoritmos de Planificación de CPU"

**Luis Ángel Reyes Frausto, Samuel Iván Sánchez Salazar**

**Resumen:** Uno de los principales objetivos de un sistema operativo es usar de manera eficiente los recursos, esto se logra con una buena administración del procesador, es decir, asignar y desasignar procesos bajo políticas que aprovechen de la mejor manera la CPU, al mismo tiempo que minimizan tiempos de retorno y espera. Para poder lograr ese rendimiento se hace uso de algoritmos de planificación como *FCFS (First Come, First Served), SJF (Shortest Job First), RR (Round Robin)* o *MLQ (Multilevel Queue).* Cada uno de estos algoritmos tiene sus ventajas y desventajas, por lo que elegir cuál usar depende del contexto en el que se aplicará y la clase de procesos con los que se trabajará. Hemos elegido dos de ellos para simularlos y comparar su desempeño en rendimiento, latencia, utilización de la CPU, tiempo de retorno, tiempo de respuesta y tiempo de espera.

**Palabras clave:** Sistema Operativo, Planificación de CPU, FCFS, RR.

## I. Introducción

En la multiprogramación se busca continuamente tener más de un programa en ejecución, aprovechando mejor los recursos pero requiriendo también una buena administración de recursos. Tanto el administrador de procesos como el administrador del procesador son los principales responsables de lograr este objetivo.

A través del administrador de procesos y el del procesador los trabajos que se encuentren listos son asignados a un núcleo de la CPU para que sean ejecutados. Por lo tanto, la cantidad de recursos utilizados y el rendimiento del sistema en general depende directamente de la correcta planeación de procesos, así como de su respectivo procesador.

Existen tres tipos de planificación:

*A.* *A largo plazo.* Podemos llamarlo también como planificador de trabajos y determina qué procesos pueden ingresar a la cola de listos. [2]
*B.* *A mediano plazo.* Maneja los procesos en la memoria, determina cuáles agregar y cuáles retirar de la memoria principal. [2]
*C.* *A corto plazo.* También llamado planificador de CPU, determina qué proceso ejecutar, asignándole un núcleo de la CPU. [2]

Los algoritmos de planificación influyen en el desempeño del planificador de CPU y se basan principalmente en criterios como la eficiencia, la utilización de la propia CPU, tiempo de respuesta, tiempo de retorno y tiempo de espera. [3]

Al comparar diferentes algoritmos de planificación, buscamos uno que se pueda aplicar a la mayoría de casos con un rendimiento óptimo.

## II. Objetivos de la planificación

Todo algoritmo de planificación busca cumplir algunos de los siguientes objetivos:

*A.* *Ser justo.* Tratar de igual manera a procesos que compartan las mismas características. [4]
*B.* *Maximizar el rendimiento.* Completar el mayor número de procesos posible por unidad tiempo. [4]
*C.* *Ser predecible.* Un mismo trabajo debe tardar aproximadamente el mismo tiempo en completarse, sin importar la carga del sistema. [4]
*D.* *Minimizar la sobrecarga.* El tiempo que el algoritmo pierde en "burocracia" debe mantenerse al mínimo. [4]
*E.* *Equilibrar el uso de recursos.* Favorecer procesos que emplean recursos subutilizados. Penalizar a los que pelean por uno sobreutilizado. [4]

F. *Evitar la postergación indefinida.* Aumentar la prioridad de procesos viejos. [4]
G. *Favorecer el "uso esperado" del sistema.* Maximizar la prioridad de los procesos que sirvan a solicitudes de usuarios interactivos. [4]
H. *Dar preferencia a los procesos que podrían causar bloqueo.* Si un proceso de baja prioridad está usando un recurso que otros procesos esperan, favorecer que termine de usarlo más rápido. [4]
I. *Favorecer a los procesos con un comportamiento deseable.* Penalizar aquellos procesos que causan muchas demoras. [4]
J. *Degradarse suavemente.* Responder con la menor penalización a los procesos preexistentes al momento de exceder el 100% de utilización del procesador. [4]

## III. Criterios de planificación

Las características propias de cada algoritmo de planificación influye en su desempeño según las propiedades de los procesos con los que se trabaja. Para generalizar la comparación, se tomarán en cuenta los siguientes criterios para evaluar cada uno de los algoritmos de planificación:

A. *Utilización de la CPU*. Maximizar su uso, buscar que la CPU esté tan ocupada como sea posible. [2]
B. *Eficiencia*. La cantidad de procesos completados por unidad de tiempo. Entre mayor sea, mejor. [2]
C. *Tiempo de retorno.* Tiempo entre que el proceso se carga y termina. Entre menor sea, mejor. [2]
D. *Tiempo de espera.* Suma del tiempo que un proceso estuvo en la cola de listos. Entre menor sea, mejor. [2]
E. *Tiempo de respuesta.* Tiempo entre que el proceso es cargado y el momento en que devuelve la primera respuesta. Entre menor sea, mejor. [2]

El proceso de la planificación consiste en determinar el orden en el que los procesos que se encuentran dentro de la cola de listos serán asignados a un núcleo de la CPU para su ejecución.

Hay que tomar en cuenta que la cola de listos no siempre se comporta como una cola *FIFO*, puede ser también una cola con prioridad o un árbol de procesos, por lo que la asignación sólo se ve afectada por los criterios del algoritmo que se utilice para la planificación.

## IV. Algoritmos de planificación

El planificador de CPU entra en acción cuando un proceso:

A. Pasa de ejecución a espera (p. ej. por solicitar una operación de E/S).
B. Pasa de ejecución a listo (p. ej. al ocurrir la interrupción de un evento externo).
C. Pasa de espera a listo (p. ej. al finalizar la operación de E/S que solicitó).
D. Termina.

Dependiendo de los casos en los que planifique, un algoritmo puede ser no expropiativo, si sólo lo hace en los casos 1 y 4, o expropiativo, en cualquier otro caso.

## V. Algoritmos simulados

De todos los algoritmos de planificación que existen, hemos elegido los siguientes dos para llevar a cabo las simulaciones y las evaluaciones:

A. *First Come, First Served (FCFS).* Básicamente es una cola FIFO, donde conforme llegan los procesos a la cola de listos, se les asigna un núcleo de la CPU para que puedan ser ejecutados. Es el más fácil de implementar, pero tiene un promedio de espera bastante elevado. Además, se trata de un algoritmo no expropiativo. [2]
B. *Round Robin (RR).* A cada proceso se le brinda un intervalo de tiempo (*quantum)* para usar el procesador, cuando consume su tiempo vuelve al final de la cola. Se puede entender como un *FCFS* con una cola circular, donde se avanza al siguiente proceso después del *quantum*. Sigue siendo relativamente simple de implementar, pero si un trabajo es muy largo y el *quantum* es muy pequeño puede llegar a desperdiciar tiempo. [2][3]

## VI. Modelo de simulación

Nuestra implementación consiste en un programa en el que se tendrá que ingresar el número de procesos que se quieren crear y tras esto, se generará un *burst time* aleatorio para

cada uno de ellos en el rango de 1-10 segundos. Por lo que inmediatamente después de ingresar el número de procesos se generará la siguiente tabla.

```
Ingrese el numero de procesos (maximo 100):
5

                              RR
Processes       Burst time      Waiting time    Turnaround time
    1               5               0               5
    2               7               18              25
    3               10              20              30
    4               8               25              33
    5               3               20              23
-----------------------------------------------------------------
 - Average waiting time = 16.6
 - Average turnaround time = 23.2
 - Average throughput time = 0.151515
_________________________________________________________________


                              FCSF
Processes       Burst time      Waiting time    Turnaround time
    1               5               0               5
    2               7               5               12
    3               10              12              22
    4               8               22              30
    5               3               30              33
-----------------------------------------------------------------
 - Average waiting time = 13.8
 - Average turnaround time = 20.4
 - Average throughput time = 0.151515
_________________________________________________________________
```

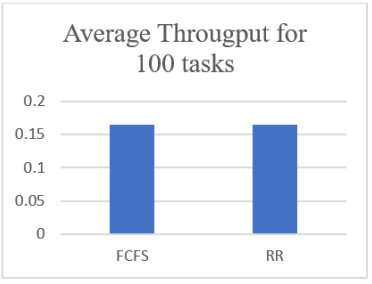
Posteriormente, el *burst time* podrá ser modificado ingresando el ID del proceso que se quiere modificar y dando el nuevo valor para volver a calcular todos los datos y hacer una comparación. De la misma manera el *quantum time* utilizado en el algoritmo *Round Robin* podrá ser modificado, estando como predeterminado 5 segundos. Estás modificaciones se pueden realizar cuantas veces se quiera, como se muestra a continuación:

```
El burst time ha sido calculado aleatoriamente
Desea cambiar el burst time de algun proceso? (s/n)
s
Ingrese el ID del proceso a modificar:
3
Process 3 burst time: 10
Ingrese el nuevo burst time:
17
Desea cambiar el burst time de algun proceso? (s/n)
n

Default time quantum: 5
Desea cambiar el time quantum? (s/n)
s
Ingrese el nuevo time quantum:
2
Desea cambiar el time quantum? (s/n)
n

                              RR
Processes       Burst time      Waiting time    Turnaround time
    1               5               15              20
    2               7               20              27
    3               17              23              40
    4               8               23              31
    5               3               16              19
-----------------------------------------------------------------
 - Average waiting time = 19.4
 - Average turnaround time = 27.4
 - Average throughput time = 0.125
_________________________________________________________________


                              FCSF
Processes       Burst time      Waiting time    Turnaround time
    1               5               0               5
    2               7               5               12
    3               17              12              29
    4               8               29              37
    5               3               37              40
-----------------------------------------------------------------
 - Average waiting time = 16.6
 - Average turnaround time = 24.6
 - Average throughput time = 0.125
_________________________________________________________________
```

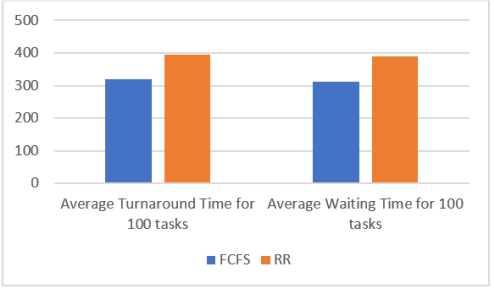
Es importante mencionar que dicha implementación solo cuenta con la simulación de un núcleo, ya que al intentar trabajar con múltiples núcleos nos encontramos con complicaciones y falta de experiencia en el uso de *threads*. Tampoco consideramos el *arrival time* de los procesos, es decir interpretamos a todos los procesos con un tiempo de llegada de 0 para no tener mayores complicaciones. Por último, pusimos un máximo de 100 procesos para hacer una correcta comparación con los resultados del artículo *"Simulation and Performance Evaluation of CPU Scheduling Algorithms"*.

## VII. Análisis de resultados

Lo primero que podemos notar tras el análisis de la eficiencia promedio para 100 procesos, es que evidentemente, al ser solo un núcleo el resultado es básicamente el mismo para ambos algoritmos como lo propone el artículo.

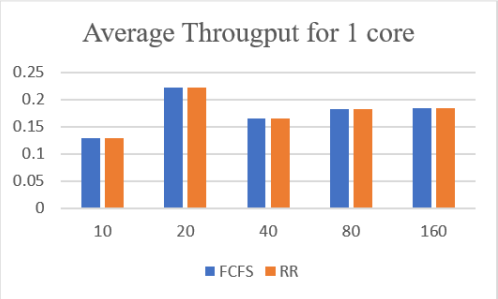



Los resultados también coinciden, en el tiempo de retorno promedio y tiempo de espera promedio, los cuales son muy similares, ya que en el *RR* es considerablemente mayor que el *FCFS* trabajando con un núcleo, aunque en nuestro caso, la diferencia entre ambos es menor, siendo el *RR* 24% más que el *FCFS* en ambos criterios, mientras que en el artículo es el aproximadamente 44% más.

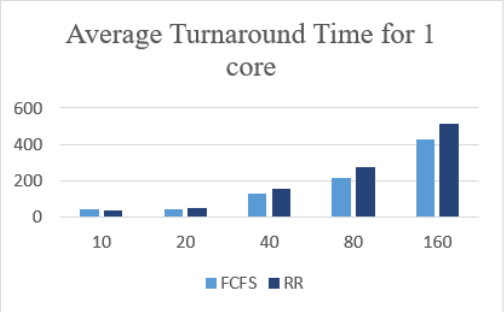



Después, simulamos con distintas cantidades de procesos (10, 20, 40, 80, 160) utilizando

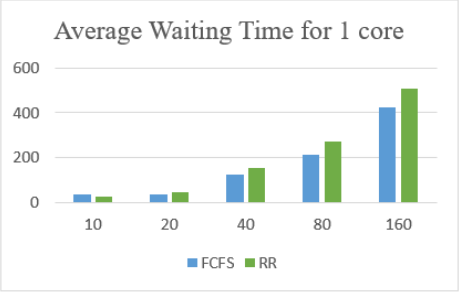
solo un núcleo. En la eficiencia promedio podemos notar que básicamente ambos algoritmos ejecutan la misma cantidad de procesos en el mismo tiempo, y realmente no existe un patrón conforme incrementa el número de procesos, sólo el factor aleatorio que en ocasiones da procesos más largos que otros.




En el tiempo de retorno promedio podemos notar que a pesar de trabajar con un solo núcleo, la gráfica es muy parecida a la del artículo, ya que conforme aumentan la cantidad de procesos, el algoritmo *RR* se torna cada vez más ineficiente en comparación con el *FCFS*, superando las 500 unidades de tiempo, aunque con un crecimiento menos exponencial de ambas partes.




El caso del tiempo de espera promedio es muy similar al tiempo de retorno, y sigue la misma trayectoria de mayor crecimiento del lado del *RR*, que la hace más ineficiente.




Es importante recalcar que esto sucede así porque los procesos tienen un *burst time* con una media similar, es decir que si se presentara alguno con un tiempo mucho mayor al promedio, es aquí donde el *RR* tomaría ventaja sobre el *FCFS* en cuanto al tiempo de espera y el tiempo de retorno.

## VIII. Conclusiones

Después de implementar y simular el desempeño de los algoritmos *FCFS* y *RR* podemos ver que si se está trabajando sólo con un núcleo para todos los procesos no hay diferencia de eficiencia entre usar un *FCFS* o un *RR*.

Esto es bastante lógico puesto que son bastante similares en implementación, siendo *FCFS* una cola *FIFO,* mientras que *RR* es una cola circular. Con un núcleo se ejecuta un proceso a la vez, lo que hace que ambas colas se comporten igual, sólo con diferencias en tiempos de espera o retorno.

Ahora bien, notamos un elemento clave que puede hacer tiempos de espera y retorno más grandes en el *RR* comparado con el *FCFS*: el *quantum time*. Si un proceso llega a ser muy grande y el *quantum time* muy pequeño, tendrá que estar constantemente esperando su turno para la CPU. Por ello, es de vital importancia definir un *quantum time* que permita reducir esos tiempos.

Ahora comparando nuestros resultados con los del artículo *"Simulation and Performance Evaluation of CPU Scheduling Algorithms"*, obtuvimos en general mediciones similares, si bien los porcentajes o magnitudes varían, el comportamiento de los algoritmos fue igual.

Aunque no pudimos llevar a cabo la simulación de los algoritmos con múltiples núcleos, si revisamos el artículo antes mencionado, podemos comprobar que los problemas del algoritmo *RR* se siguen presentando, lo que nos puede llevar a concluir que no influye la cantidad de núcleos a la hora de comparar estos dos algoritmos. Siendo *FCFS,* el mejor algoritmo en casi todos los casos.

## Referencias

[1] A. Silberschatz, P. Galvin y G. Gagne,

Operating System Concepts, New Jersey: Wiley, 2018.

[2] M. Rodríguez, «Administración de procesos,» 2023. [En línea]. Available: https://www.canva.com/design/DAFasKJijCc/view?utm_content=DAFasKJijCc&utm_campaign=designshare&utm_medium=embeds&utm_source=link#2. [Último acceso: 27 Febrero 2023].

[3] S. Almakdi, M. Aleisa y M. Alshehri, "Simulation and Performance Evaluation of CPU Scheduling Algorithms", International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, n.º 3, p. 6, 2015.

[4] G. Wolf, E. Ruiz, F. Bergero y E. Meza. "Planificación de procesos". http://sistop.gwolf.org/pdf/04_planificacion_de_procesos.pdf (accedido el 27 de febrero de 2023)