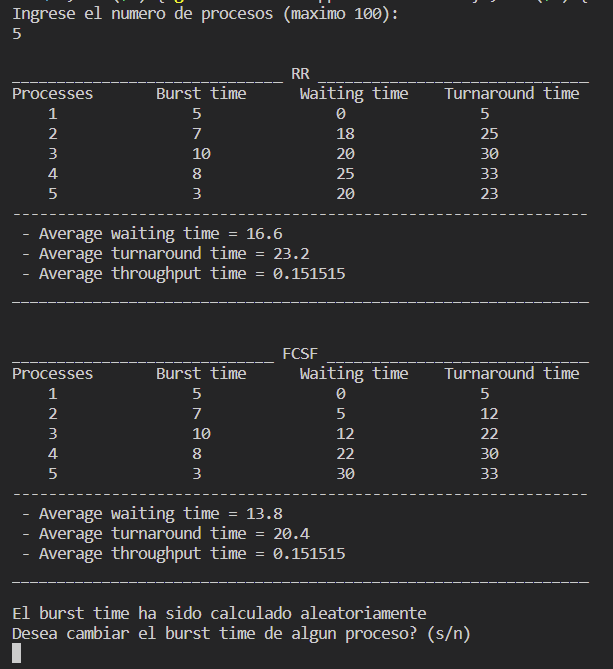
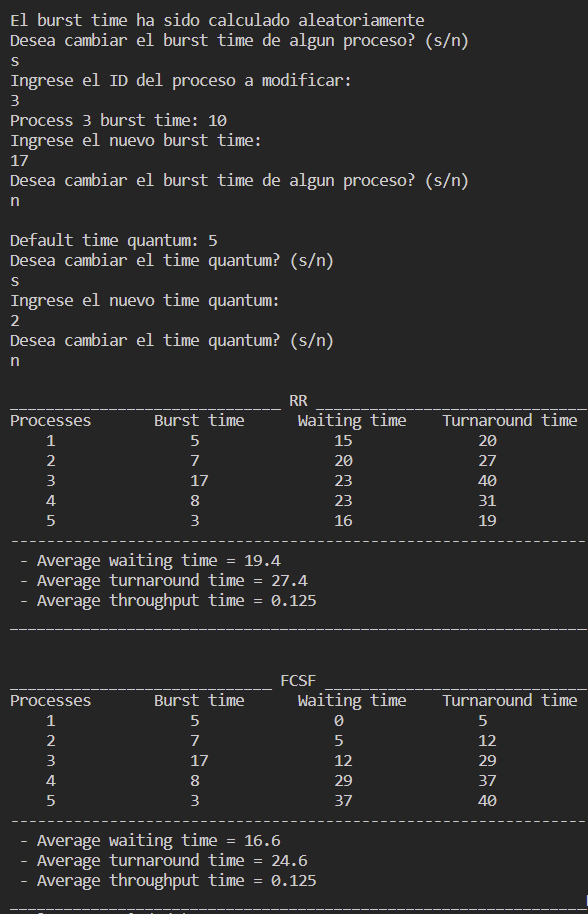
Nuestra implementación consiste en un programa en el que se tendrá que ingresar el numero de procesos que se quieren crear y tras esto, se generará un burst time aleatorio para cada uno de ellos en el rango de 1-10 segundos. Por lo que inmediatamente después de ingresar el numero de procesos se generará la siguiente tabla.



Posteriormente, el burst time podrá ser modificado ingresando el id del proceso que se quiere modificar y dándolo 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 default a los 5 segundos. Estás modificaciones se pueden realizar cuantas veces se quiera, como se muestra a continuación:



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.

RESULTADOS

Lo primero que podemos notar tras el análisis del average througput 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 average turnaround y average waiting time, 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 articulo es el aproximadamente 44% más.

Después, simulamos con distintas cantidades de procesos (10, 20, 40, 80, 160) utilizando solo un núcleo. En el Average Throughput podemos notar que básicamente ambos algoritmos ejecutan la misma cantidad de procesos en el mismo tiempo, y realmente no teniendo un patron conforme incrementa el numero de procesos más que el factor aleatorio que en ocasiones da procesos más largos que otros.

En el average turnaround time 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 average waiting time es muy similar al average turnaround time, y sigue la misma trayectoria de mayor crecimiento del lado del RR, que repito, la hace más ineficiente. Pero también es importante recalcar que esto sucede así porque los procesos tiene 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 waiting time y el turnaround time