Trabajo Práctico 4
Procesos – Scheduling
Multiprocesamiento

1.- Responda en forma sintética sobre los siguientes conceptos:

a) Programa y Proceso.

Un programa es un conjunto de instrucciones que una vez ejecutadas realizarán una o varias tareas en una computadora. Es estático, no tiene Program Counter (PC) y existe desde que se edita hasta que se borra.

Un proceso es un programa en ejecución gestionado por el Sistema Operativo (SO). Es dinámico, tiene PC y su ciclo de vida comprende desde que se lo "dispara" hasta que se termina de ejecutar.

b) Defina Tiempo de retorno (TR) y Tiempo de espera (TE) para un Job.

Tiempo de retorno: Tiempo que transcurre entre que el proceso llega al sistema hasta que completa su ejecución.

Tiempo de espera: Tiempo que el proceso se encuentra en el sistema esperando (sin ejecutarse) (TR – Tcpu).

c) Defina Tiempo Promedio de Retorno (TPR) y Tiempo promedio de espera (TPE) para un lote de JOBS

Los tiempos promedios son los promedios de los anteriores.

d) ¿Qué es el Quantum?

El quantum es el tiempo dado a un proceso para ejecutarse.

e) ¿Qué significa que un algoritmo de scheduling sea apropiativo o no apropiativo (Preemptive o Non-Preemptive)?

Nonpreemptive

Una vez que un proceso esta en estado de ejecución, continua hasta que termina o se bloquea por algún evento (por ej. I/O).

Preemptive

El proceso en ejecución puede ser interrumpido y llevado a la cola de listos por el SO.

Mayor overhead pero mejor servicio

Un proceso no monopoliza el procesador.

f) ¿Qué tareas realizan?

Short Term Scheduler: determina qué proceso listo se ejecuta.

Long Term Scheduler: Admite nuevos procesos a memoria. Controla el *grado de multiprogramación*, es decir, la cantidad de procesos en memoria.

Puede no existir este scheduler y absorber esta tarea el de short term.

Medium Term Scheduler: Realiza Swapping (intercambio) entre disco y memoria cuando el SO lo determina.

Si es necesario, reduce el grado de multiprogramación. Saca temporariamente de memoria los procesos que sea necesario para mantener el equilibrio del sistema.

g) ¿Qué tareas realiza el Dispatcher?

Hace cambio de contexto, cambio de modo de ejecución..."despacha" el proceso elegido por el short term (es decir, "salta" a la instrucción a ejecutar).

Loader: carga en memoria el proceso elegido por el long term.

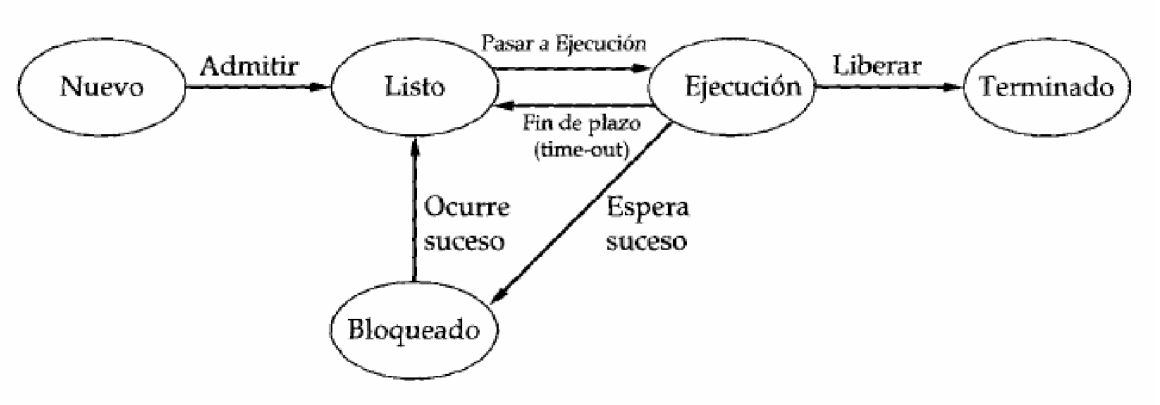
2.- Procesos:

a) ¿Cuál es la información mínima que el SO debe tener sobre un proceso? ¿En que estructura de datos asociada almacena dicha información?

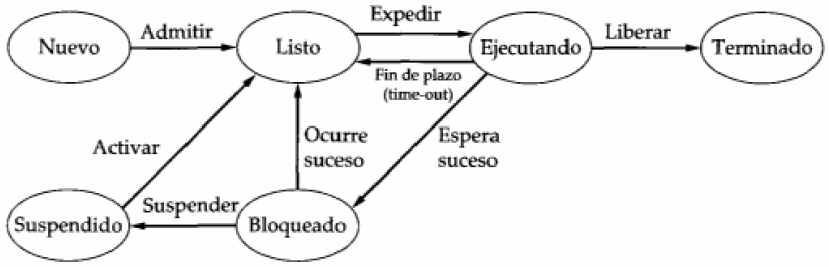
La información mínima que el SO debe tener sobre un proceso es el "contexto de un proceso", el cual contiene los registros de la CPU (inclusive el PC), la prioridad del proceso, si tiene E/S pendientes, etc. Esta información se almacena en la memoria principal.

b) ¿Qué significa que un proceso sea "CPU Bound" y "I/O Bound"?
Son formas de clasificar los procesos según su historial de ejecución:
Un proceso CPU bound significa ligado a la CPU (mayor tiempo utilizando la CPU) y un proceso I/O Bound que está ligado a Entrada/Salida (mayor tiempo esperando por E/S).

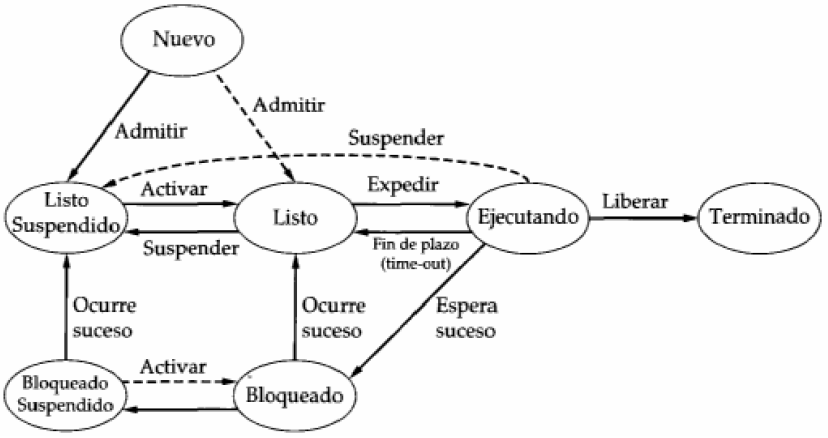
c) ¿Cuáles son los estados posibles por los que puede atravesar un proceso?
Ejecución: El proceso que está actualmente en ejecución.
Listo: Proceso que está preparado para ejecutar, en cuanto se le dé la oportunidad.
Bloqueados: Proceso que no puede ejecutar hasta que se produzca cierto suceso, como la terminación de una operación de E/S.
Nuevo: Proceso que se acaba de crear, pero que aún no ha sido admitido por el sistema operativo en el grupo de procesos ejecutables.
Terminado: Un proceso que ha sido excluido por el sistema operativo del grupo de procesos ejecutables, bien porque se detuvo o porque fue abandonado por alguna razón.



d) Explique mediante un diagrama las posibles transiciones entre los estados.



(a) Con un estado Suspendido



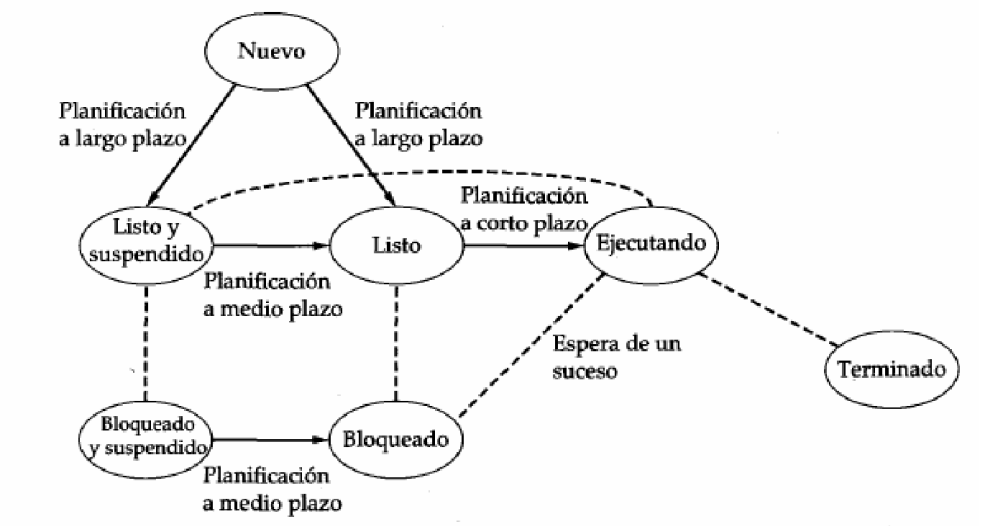
Bloqueado → Bloqueado y suspendido: Si no hay procesos Listos, entonces al menos un proceso Bloqueado se expulsa para dar lugar a otro proceso que no esté bloqueado. Esta transición puede hacerse

aún cuando hay procesos listos disponibles, cuando el sistema operativo determina que el proceso que está actualmente en Ejecución o un proceso Listo que sería conveniente expedir, requiere más memoria principal para mantener un rendimiento adecuado.
Bloqueado y suspendido → Listo y suspendido: Un proceso en estado Bloqueado y sus-pendido se pasa al estado Listo y suspendido cuando ocurre el suceso que estaba esperando. Nótese que esto requiere que esté accesible para el sistema operativo la información relativa a los procesos Suspendidos.
Listo y suspendido → Listo: Cuando no hay procesos Listos en la memoria principal, el sistema operativo tendrá que traer uno para continuar la ejecución. Además, puede darse el caso de que un proceso en estado Listo y suspendido tenga una prioridad mayor que la de un proceso en estado Listo. En tal caso, el diseñador del sistema operativo puede decidir que es más importante tomar el proceso de mayor prioridad que minimizar el intercambio.
Listo → Listo y suspendido: Generalmente, el sistema operativo prefiere suspender a un proceso Bloqueado en vez de a uno Listo, ya que el proceso Listo podría ejecutarse de inmediato, mientras que el proceso Bloqueado estará ocupando espacio en la memoria principal sin poder ejecutarse. Sin embargo, puede ser necesario suspender un proceso Listo si ésta es la única forma de liberar un bloque lo suficientemente grande de memoria principal. Por último el sistema operativo puede escoger suspender un proceso Listo de más baja prioridad en lugar de uno Bloqueado que sea de prioridad más alta si él cree que el proceso Bloqueado pronto estará listo.

Otras transiciones son también dignas de consideración:

Nuevo → Listo, Suspendido y Nuevo → Listo: Cuando se crea un nuevo proceso, se le puede añadir a la cola de listos o a la de listos y suspendidos. En ambos casos, el sistema operativo necesita construir unas tablas para poder administrar el proceso y asignarle un espacio de direcciones. Podría ser preferible que el sistema operativo llevara a cabo estas labores en un primer momento, de modo que se mantuviera una reserva grande de procesos que no están bloqueados. Con esta estrategia sería frecuente el caso de que hubiese poco espacio en memoria principal para un nuevo proceso; de ahí el uso de la nueva transición Nuevo → Listo y suspendido. Por otro lado, puede argumentarse que una filosofía de creación de los procesos "justo a tiempo", retrasando la creación todo lo que se pueda, reduciría la sobrecarga del sistema operativo y le permitiría llevar a cabo las tareas de creación de procesos en el momento en el que el sistema esté atascado de todas maneras con procesos Bloqueados.
Bloqueado y suspendido → Bloqueado: La inclusión de esta transición puede parecer resultado de un mal diseño. Después de todo, si un proceso no está listo para ejecutarse y aún no está en memoria principal, ¿cuál es el interés por traerlo a memoria? Pero la siguiente situación es posible: Un proceso termina, liberando memoria principal. Hay un proceso en la cola de Bloqueados y suspendidos que tiene una prioridad mayor que la de cualquier proceso de la cola de Listos y suspendidos, así que el sistema operativo tiene razones para suponer que pronto ocurrirá el suceso por el que el proceso está bloqueado. En estas circunstancias, podría parecer razonable traer un proceso Bloqueado a memoria antes que un proceso Listo.
Ejecución → Listo y suspendido: Generalmente, un proceso en Ejecución pasa al estado Listo cuando expira su fracción de tiempo asignado. Sin embargo, si se está expulsando al proceso porque hay un proceso de prioridad mayor en la lista de Bloqueados y suspendidos que se acaba de desbloquear, entonces el sistema operativo podría pasar el proceso en Ejecución directamente a la cola de Listos y suspendidos, liberando espacio en la memoria principal.
Varios → Terminado: Normalmente, los procesos terminan mientras están ejecutándose, bien porque se completaron o bien por causa de alguna condición drástica de error. Sin embargo, en algunos sistemas operativos, un proceso puede ser finalizado por el proceso que lo creó o bien finalizar cuando termina el proceso padre. Si se permite esto, un proceso situado en cualquier estado podrá pasar al estado Terminado.

e) ¿Que scheduler de los mencionados en 1f) se encarga de las transiciones?



3.- Para los siguientes algoritmos de scheduling:

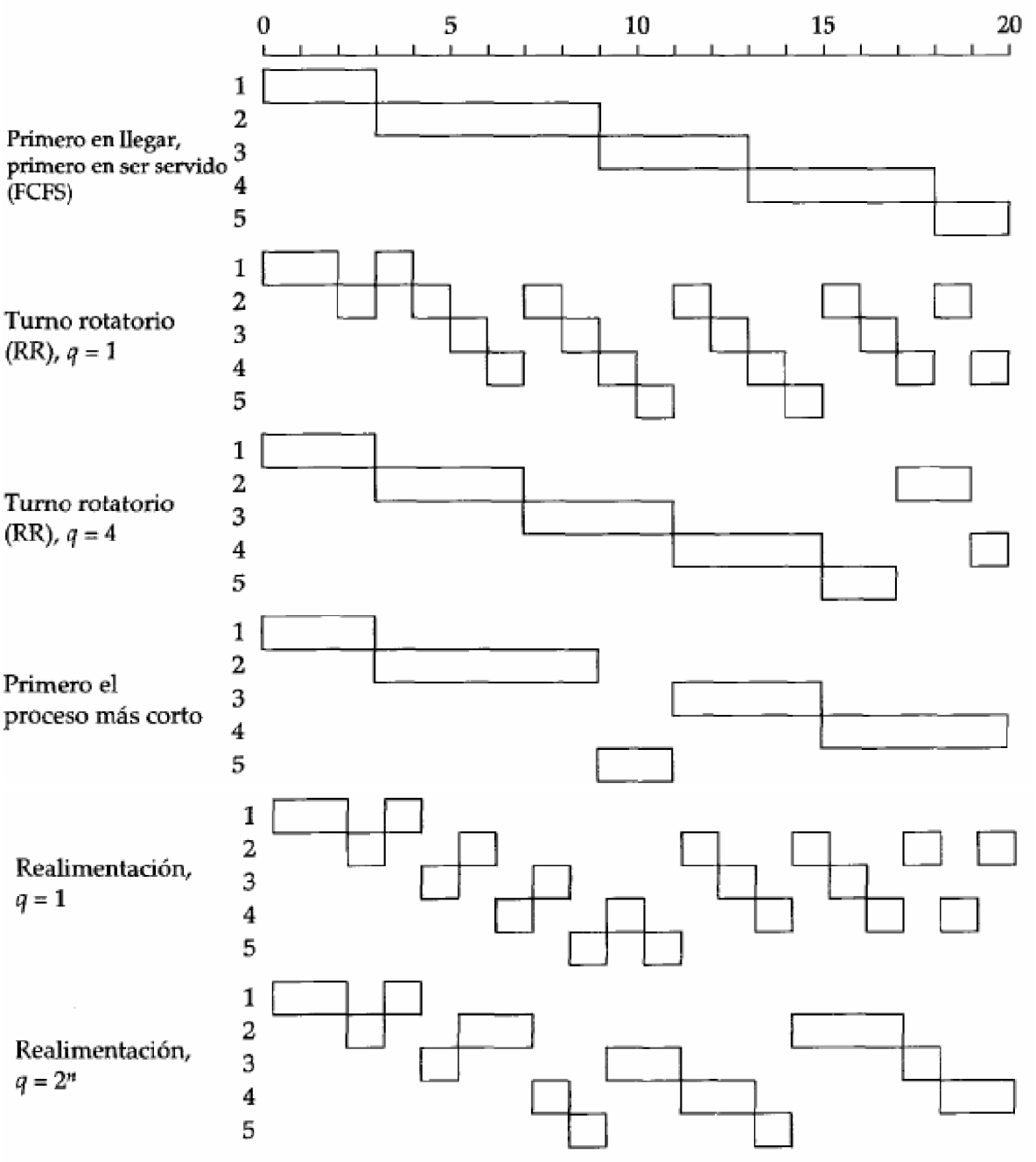
FCFS (Fisrt Coome First Served)
SJF (Shortest Job First)
Round Robin
Prioridades

a) Explique su funcionamiento mediante un ejemplo.

First-Come-First-Served (FCFS)
Cada proceso se coloca en la cola de listos. Cuando hay que elegir un proceso para ejecutar, se selecciona el más viejo en la cola de listos (FIFO). Este algoritmo favorece a procesos "CPU Bound".
Los procesos con carga de I/O deben esperar a que los procesos con carga de procesador se completen.
Luego, el proceso con carga de I/O rápidamente libera el procesador y se bloquea nuevamente.
Shortest Job First (SJF)
Es una Política nonpreemptive que selecciona el proceso mas corto primero. Los procesos cortos se colocan delante de procesos largos. Los procesos largos pueden sufrir starvation (Inanición).
Round Robin (RR)
Preemption (apropiación) basada en un reloj.
Quantum: Medida que determina cuánto tiempo podrá usar el procesador cada proceso.
Cuando un proceso es expulsado de la CPU, es colocado al final de la Ready Queue (cola de listos) y se selecciona otro (FIFO Circular).
Cada proceso se ejecuta durante una fracción de tiempo QUANTUM (Q)
Existe un "contador" que indica las unidades de CPU en las que se ejecutó. Cuando el mismo llega a 0 (cero) el proceso es expulsado.
Existen 2 variantes con respecto al valor inicial del "contador" cuando un proceso es asignado a la CPU:
TIMER VARIABLE:
El "contador" se inicializa en Q cada vez que un proceso es asignado a la CPU.
Este Esquema es el más utilizado en los algoritmos RR, así como también por el simulador.
TIMER FIJO:
El "contador" solo se inicializa a Q cuando su valor es 0 (cero). Es como si el "contador" se compartiera entre los procesos.
Uso de Prioridades
Cada proceso tiene un valor que representa su prioridad. El scheduler selecciona el proceso de mayor prioridad de los que se encuentran en la Ready Queue.
Para simplificar, existe una Ready Queue por cada nivel de prioridad.
Los procesos de Baja Prioridad pueden sufrir starvation (Inanición). La solución consiste en permitir a un proceso cambiar su prioridad durante su ciclo de vida.
Puede ser preemptive.

Ejemplo:

Proceso	Instante de Llegada	Tiempo de Servicio
1	0	3
2	2	6
3	4	4
4	6	5



b) ¿Alguno de ellos requiere algún parámetro para su funcionamiento?
Round Robin requiere el quantum necesario y Prioridades necesita saber la prioridad de los procesos.

c) Cual es el mas adecuado según los tipos de procesos y/o SO. d) Cite ventajas y desventajas de su uso.

Depende de muchos factores, por ejemplo, si los procesos son orientados a CPU o a E/S, la longitud de las ráfagas de los procesos y la finalidad del sistema operativo. El algoritmo FCFS tiene a favorecer a los procesos ligados a la CPU y cortos. El turno rotatorio es particularmente efectivo en sistemas de propósito general y de tiempo compartido o de proceso de transacciones. Una desventaja del turno rotatorio es el tratamiento que hace de los procesos con carga de procesador y con carga de E/S. Generalmente, un proceso con carga de E/S tiene ráfagas de procesador (cantidad de tiempo consumido ejecutando entre dos operaciones de E/S) más cortas que un proceso con carga de procesador. Si hay una mezcla de procesos con carga de procesador y con carga de E/S, ocurrirá lo siguiente: Un proceso con carga de E/S utiliza el procesador durante un periodo corto y después se bloquea en la E/S; espera a que se complete la operación de E/S y entonces vuelve a la cola de Listos. Por otro lado, un proceso con carga

de procesador generalmente hace uso de un cuanto de tiempo completo cuando ejecuta e, inmediatamente, retorna a la cola de Listos. Así pues, los procesos con carga de procesador tienden a recibir una porción desigual de tiempo de procesador.

El SJF es conveniente en SO de tipo batch, donde se puede estimar de antemano el tiempo de retorno de los trabajos, aunque, si no se controla, puede generar que los procesos largos mueran de inanición ante un flujo constante de procesos pequeños.

4.- Para el algoritmo Round Robin, existen 2 variantes:

Timer Fijo
Timer Variable

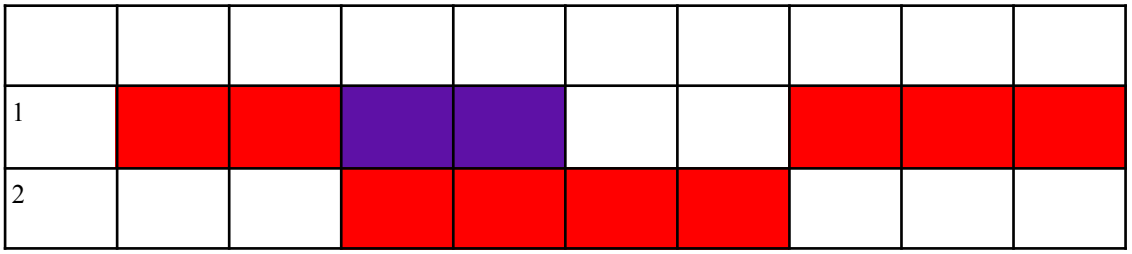
a) ¿Qué significan estas 2 variantes?

Existe un "contador" que indica las unidades de CPU en las que se ejecutó. Cuando el mismo llega a 0 (cero) el proceso es expulsado.
Existen 2 variantes con respecto al valor inicial del "contador" cuando un proceso es asignado a la CPU:
TIMER VARIABLE:
El "contador" se inicializa en Q cada vez que un proceso es asignado a la CPU.
Este Esquema es el más utilizado en los algoritmos RR, así como también por el simulador.
TIMER FIJO:
El "contador" solo se inicializa a Q cuando su valor es 0 (cero). Es como si el "contador" se compartiera entre los procesos.

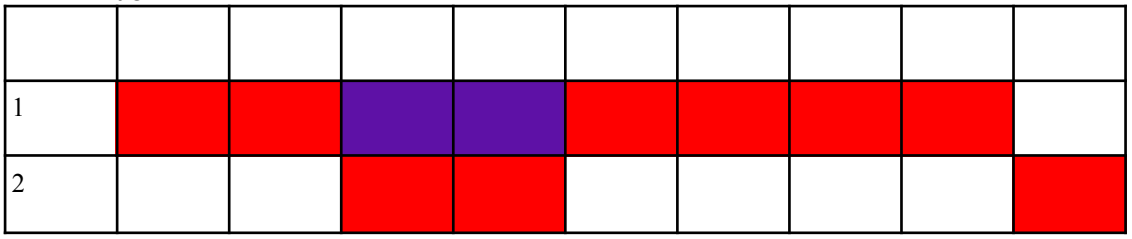
b) Explique mediante un ejemplo sus diferencias.

Supongamos que se cuenta con dos procesos uno y dos, ejecutándose en un Round Robin con un quantum de cuatro. El proceso uno se ejecuta durante dos instantes de tiempo, para luego bloquearse por una operación de E/S.

CPU
E/S
TIMER VARIABLE



TIMER FIJO



c) En cada variante ¿Dónde debería residir la información del Quantum?

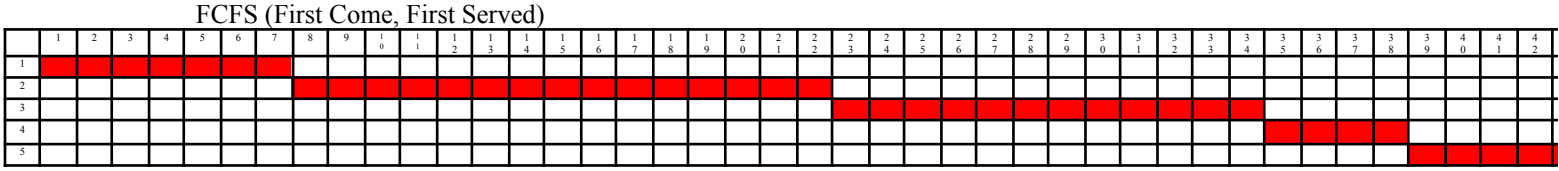
En un contador.

5.- Se tiene el siguiente lote de procesos que arriban al sistema en el instante 0 (cero)

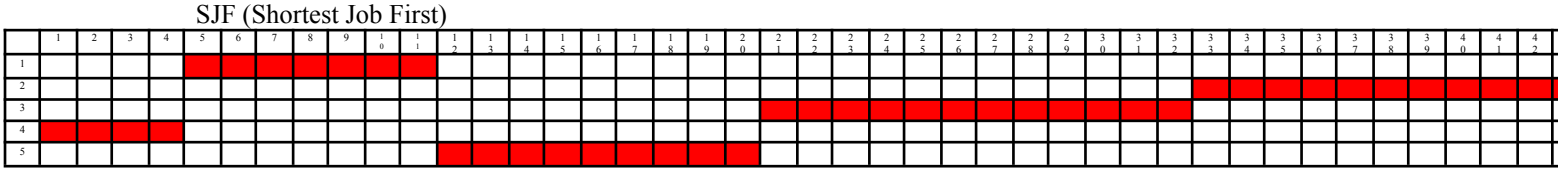
Job	Unidades de CPU
1	7
2	15
3	12
4	4
5	9

a) Realice los diagramas de Gantt según los siguientes algoritmos de Scheduling:

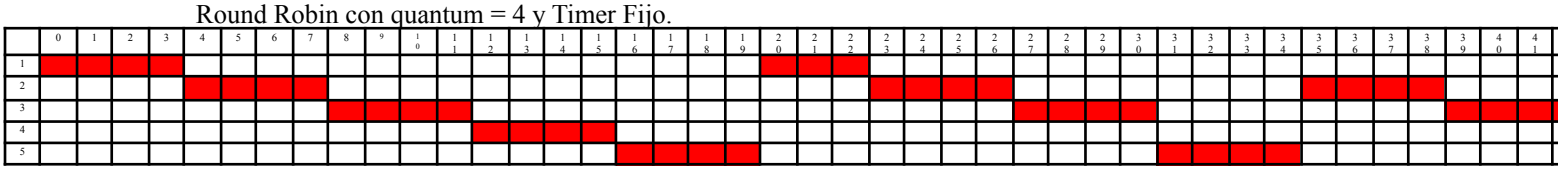
FCFS (First Come, First Served)



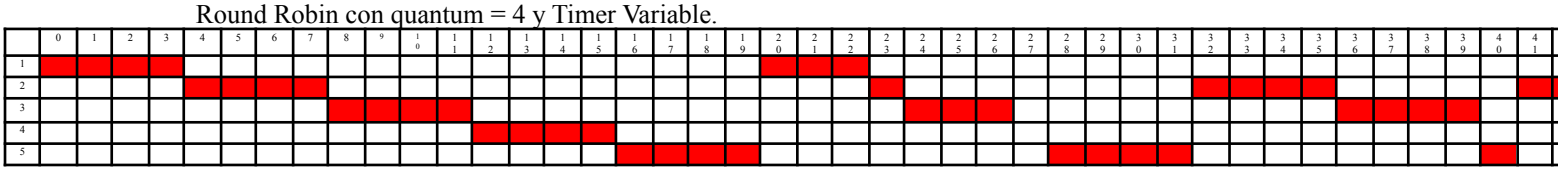
SJF (Shortest Job First)



Round Robin con quantum = 4 y Timer Fijo.



Round Robin con quantum = 4 y Timer Variable.



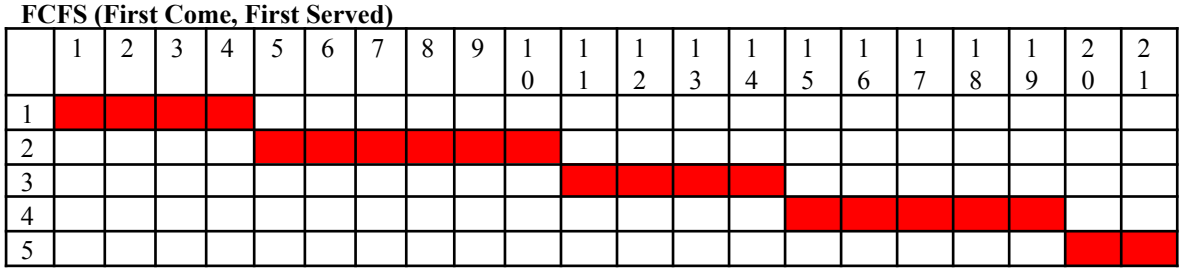
b) Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.
c) En base a los tiempos calculados compare los diferentes algoritmos.

6.- Se tiene el siguiente lote de procesos:

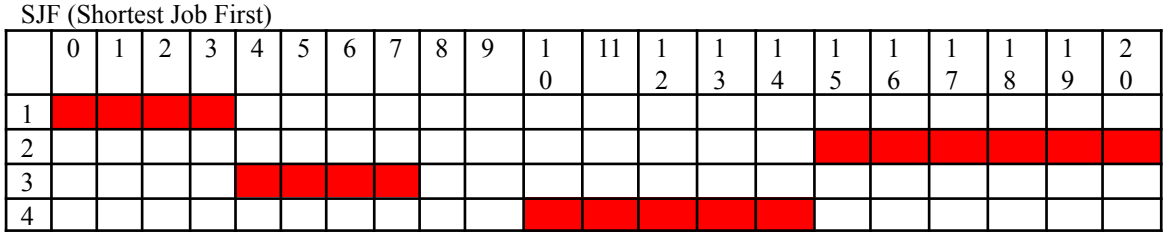
Job	Llegada	Unidades de CPU
1	0	4
2	2	6
3	3	4
4	6	5
5	8	2

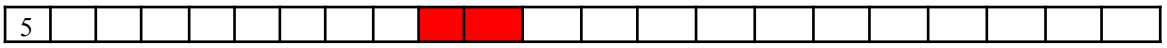
a) Realice los diagramas de Gantt según los siguientes algoritmos de Scheduling:

FCFS (First Come, First Served)

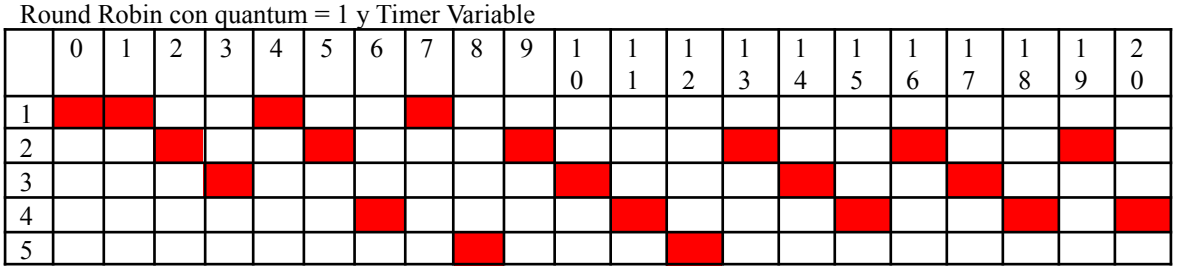


SJF (Shortest Job First)

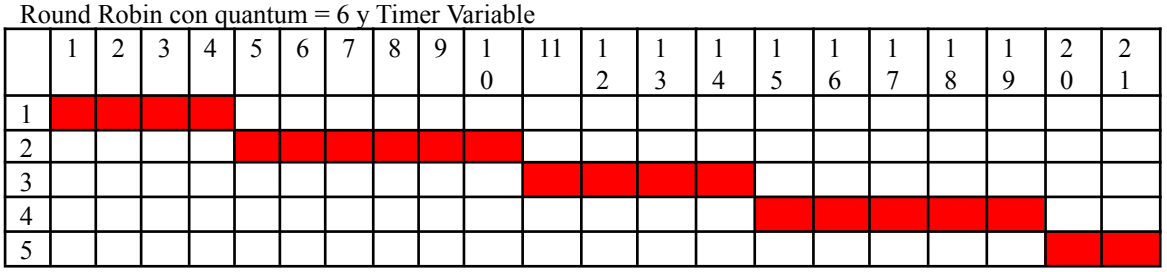




Round Robin con quantum = 1 y Timer Variable



Round Robin con quantum = 6 y Timer Variable



b) Para cada algoritmo calcule el TR y TE para cada job así como el TPR y el TPE.

FCFS (Fisrt Coome First Served):

TR:
1:4
2:8
3:11
4:13
5:13
Promedio: 9,8

TE:
1:0
2:2
3:8
4:9
5:12
Promedio: 6,2

SJF (Shortest Job First):

TR:
1:4
2:20
3:6
4:10
5:3
Promedio: 8,6

TE:
1:0
2:14
3:2
4:5
5:1
Promedio: 4,4

Round Robin Q=1:

TR:
1:8
2:17
3:14
4:14
5:5
Promedio: 11,6

TE:
1:4
2:12
3:11
4:10
5:3
Promedio: 8

Round Robin Q=6:

TR:
1:4
2:8
3:11

4:13
5:13
Promedio: 9,8
TE:

1:0
2:2
3:8
4:9
5:12
Promedio: 6,2

c) En base a los tiempos calculados compare los diferentes algoritmos.
Un RR con Q grande se comporta como un FCFS. A su vez, un RR con Q muy pequeño, tiene los peores promedios de TR y TE.
En este ejemplo, el mejor algoritmo sería SJF.

d) En el algoritmo Round Robin, qué conclusión se puede sacar con respecto al valor del quantum.
Ya lo dije.
e) ¿Para el algoritmo Round Robin, en qué casos utilizaría un valor de quantum alto y qué ventajas y desventajas obtendría?
En realidad no lo utilizaría, pero si me vas a andar obligando, sería en el caso de tener procesos con alta carga y tiempos de llegada bastante espaciados.
La ventaja de un RR con Q alto sería que un proceso con gran cantidad de uso de unidades de CPU terminaría sin tanto tiempo de espera. Pero la desventaja es evidente: si un proceso ocupa todo el tiempo del Q y llegó uno mucho antes, tendrá un TE muy alto y se produce riesgo de inanición.

7.- Una variante al algoritmo SJF es el algoritmo SJF apropiativo o SRTF (Shortest Remaining Time First). Selecciona el proceso al cual le resta menos tiempo de ejecución.

a) Realice el diagrama del Gantt para este algoritmo según el lote de trabajos del ejercicio 6).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	■	■	■	■																	
2																■	■	■	■	■	■
3					■	■	■	■													
4											■	■	■	■	■						
5									■	■											

b) ¿Nota alguna ventaja frente a otros algoritmos?
No, el algoritmo se comporta igual que un SJF porque los trabajos no tienen E/S que los bloqueen, entonces el tiempo restante es igual al tiempo completo de CPU del trabajo.

8.- Suponga que se agregan las siguientes prioridades al lote de procesos del ejercicio 6:

Job	Prioridad
1	3
2	4
3	2
4	1
5	2

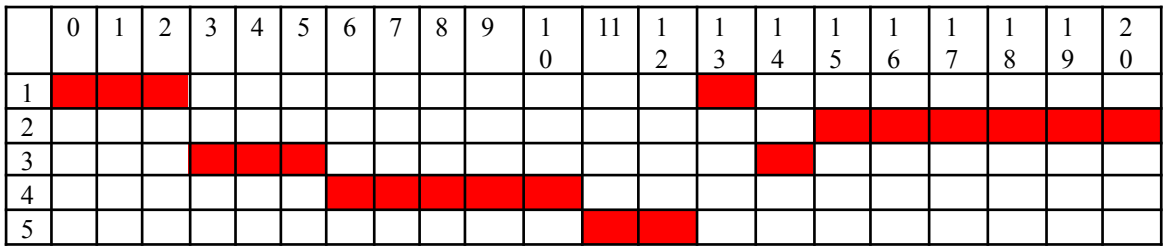
Job	Llegada	Unidades de CPU
1	0	4
2	2	6
3	3	4
4	6	5
5	8	2

donde un menor número indica mayor prioridad.
Realice el diagrama de Gantt correspondiente al algoritmo de planificación por prioridades según las variantes:

No Apropiativa

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	■	■	■	■																	
2																■	■	■	■	■	■
3					■	■	■	■													
4									■	■	■	■	■								
5														■	■						

Apropiativa



b) Calcule el TR y TE para cada job así como el TPR y el TPE.

TR No Apropiativo
1:4
2:18
3:5
4:8
5:7
Promedio: 8,4

TE No apropiativo
1:0
2:12
3:1
4:2
5:5
Promedio: 4

TR Apropiativo
1:14
2:18
3:11
4:5
5:5
Promedio: 8,6

TE Apropiativo
1:10
2:12
3:7
4:0
5:3
Promedio: 6,2

c) ¿Nota alguna ventaja frente a otros algoritmos? ¿Bajo que circunstancias lo utilizaría y ante que situaciones considera que la implementación de prioridades podría no ser de mayor relevancia?
Ventajas no encuentro, los números son parecidos, si bien los tiempos de espera disminuyen algo.

9.- Inanición (Starvation)

a) ¿Qué significa?
Es un problema donde a un proceso se le deniega siempre el acceso a un recurso compartido, por lo tanto, la tarea a ejecutar nunca puede finalizar y el proceso "muere".

b) ¿Cuál/es de los algoritmos vistos puede provocarla?
La utilización de prioridades en muchos sistemas operativos multitarea podría causar que procesos de alta prioridad estuvieran ejecutando siempre y no permitieran la ejecución de procesos de baja prioridad, causando inanición en estos. Es más, si un proceso de alta prioridad está pendiente del resultado de un proceso de baja prioridad que no se ejecuta nunca, entonces este proceso de alta prioridad también experimenta inanición.

c) ¿Existe alguna técnica que evite la inanición para el/los algoritmos mencionados en b)?
Wikipedia lo sabe:
Para evitar estas situaciones los planificadores modernos incorporan algoritmos para asegurar que todos los procesos reciben un mínimo de tiempo de CPU para ejecutarse.
Por ejemplo, un proceso puede cambiar su prioridad durante su vida, aumentando a mayor tiempo de espera, o disminuyendo a medida que ya ha recibido una determinada cantidad de CPU.

10.- Los procesos, durante su ciclo de vida, pueden realizar operaciones de I/O como lecturas o escrituras a disco, cintas, uso de impresoras, etc.

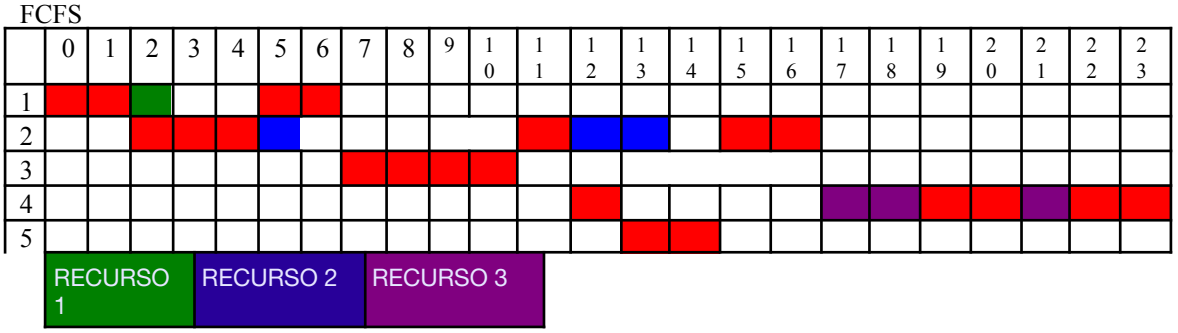
El SO mantiene para cada dispositivo, que se tiene en el equipo, una cola de procesos que espera por la utilización del mismo (al igual que ocurre con la Cola de Listos y la CPU, ya que la CPU es un dispositivo mas).
Cuando un proceso en ejecución realiza una operación de I/O, el mismo es expulsado de la CPU y colocado en la cola correspondiente a el dispositivo involucrado en la operación.
El SO dispone también de un "I/O Scheduling" que administra cada cola de dispositivo a través de algún algoritmo (FCFS, Prioridades, etc.). Si al colocarse un proceso en la cola del dispositivo, la misma se

encuentra vacía el mismo será atendido de manera inmediata, caso contrario, deberá esperar a que el SO lo seleccione según el algoritmo de scheduling establecido.
Los mecanismos de I/O utilizados hoy en día permiten que la CPU no sea utilizada durante la operación, por lo que el SO puede ejecutar otro proceso que se encuentre en espera una vez que el proceso bloqueado por la I/O se coloca en la cola correspondiente.
Cuando el proceso finaliza la operación de I/O el mismo retorna a la cola de listos para competir nuevamente por la utilización de la CPU.
Suponiendo que la cola de listos de todos los dispositivos se administra mediante FCFS, realice los diagramas de Gantt según las siguientes situaciones:

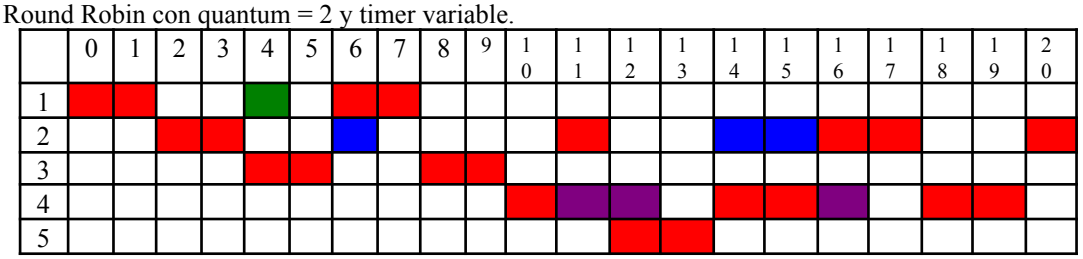
a) Suponga que al lote de procesos del ejercicio 6) se agregan las siguientes operaciones de entrada salida:

Para los siguientes algoritmos de Scheduling:

FCFS



Round Robin con quantum = 2 y timer variable.

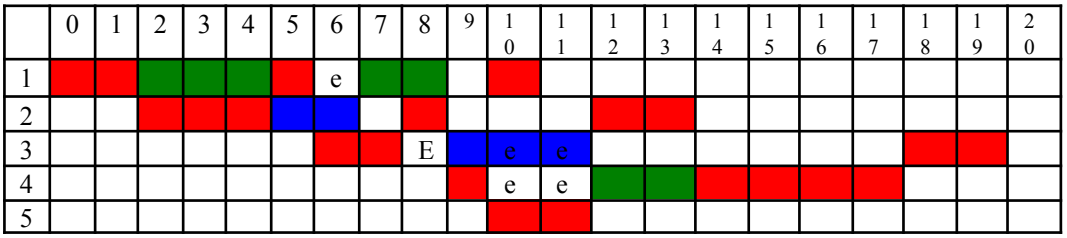


b) Suponga que al lote de procesos del ejercicio 6) se agregan las siguientes operaciones de entrada salida:

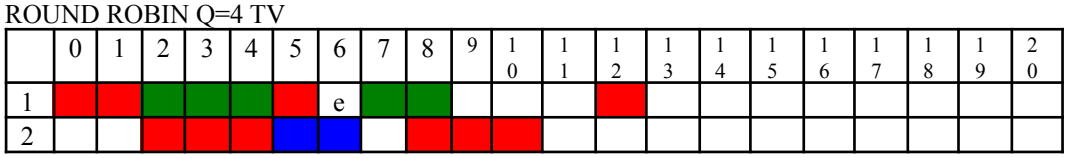
Job	I/O (recur,inst,dur)
1	(R1, 2, 3) (R1, 3, 2)
2	(R2, 3, 2)
3	(R2, 2, 3)
4	(R1, 1, 2)

Job	Llegada	Unidades de CPU
1	0	4
2	2	6
3	3	4
4	6	5
5	8	2

FCFS



ROUND ROBIN Q=4 TV



11.- Algunos algoritmos pueden presentar ciertas desventajas cuando en el sistema se cuenta con procesos ligados a CPU y procesos ligados a entrada salida. Analice las mismas para los siguientes algoritmos:

a) Round Robin

b) SRTF (Shortest Remaining Time First)

a. Round Robin

Una desventaja del turno rotatorio es el tratamiento que hace de los procesos con carga de procesador y con carga de E/S. Generalmente un proceso con carga de E/S tiene ráfagas de procesador (cantidad de tiempo consumido ejecutando entre dos operaciones de E/S) mas cortas que un proceso con carga de procesador. Si hay una mezcla de procesos con carga de procesador y con carga de E/S, ocurrirá lo siguiente: un proceso con carga de E/S utiliza el procesador durante un periodo corto y después se bloquea en la E/S; espera a que se complete la operación de E/S y entonces vuelve a la cola de Listos. Por otro lado, un proceso con carga de procesador generalmente hace uso de un cuanto de tiempo completo cuando se ejecuta e inmediatamente retorna a la cola de listos. Asi pues, los procesos con carga procesador tienden a recibir una porción desigual de tiempo de procesador, lo que origina un rendimiento pobre de los procesos con carga de E/S un mal aprovechamiento de los dispositivos de E/S y un incremento de la variabilidad del tiempo de respuesta.

b. SRTF (Shortest Remaining Time First)

Como en el SJF, el planificador debe disponer de una estimación del tiempo de proceso para poder llevar a cabo la función de selección, existiendo el riesgo de inanición para procesos largos.

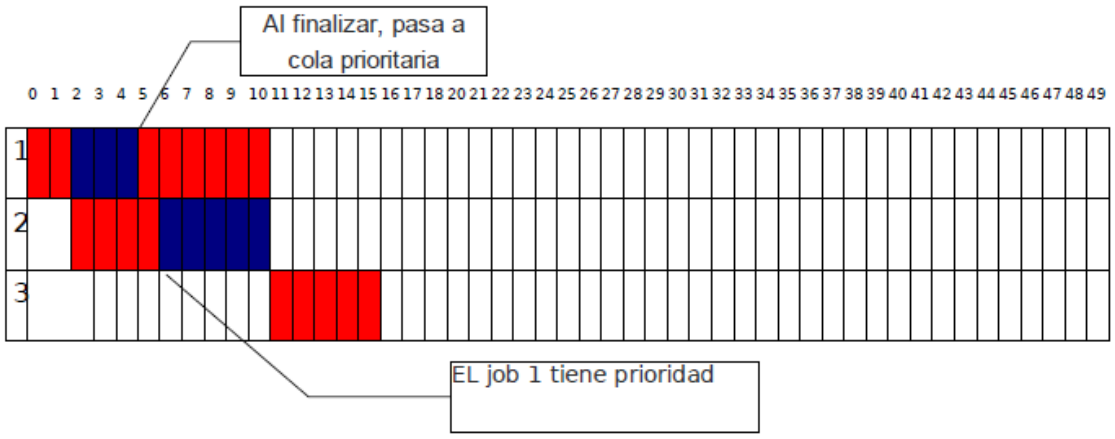
El SRT no presenta el sesgo favorable a los procesos largos del FCFS. Al contrario que el turno rotatorio, no se generan interrupciones adicionales y, así, el coste se ve reducido. Por contra, se deben los tiempos de servicio transcurridos, lo que contribuye a la sobrecarga. El SRT también debe producir unos tiempos de retorno mejores que los del SJF, puesto que los trabajos cortos reciben una atención inmediata y preferente a los trabajos largos.

12.- Para equiparar la desventaja planteada en el ejercicio 11), se plantea la siguiente modificación al algoritmo:

Algoritmo VRR (Virtual Round Robin): Este algoritmo funciona igual que el Round Robin, con la diferencia que cuando un proceso regresa de una I/O se coloca en una cola auxiliar. Cuando se tiene que tomar el próximo proceso a ejecutar, los procesos que se encuentran en la cola auxiliar tienen prioridad sobre los otros. Cuando se elije un proceso de la cola auxiliar se le otorga el procesador por tantas unidades de tiempo como le falto ejecutar en su ráfaga de CPU anterior, esto es, se le otorga la CPU por un tiempo que surge entre la diferencia del quantum original y el tiempo usado en la última ráfaga de CPU.

a) Analice el funcionamiento de este algoritmo mediante un ejemplo. Marque en cada instante en que cola se encuentran los procesos.

JOB	LLEGADA	CPU	I/O
1	0	8	(R1,2,3)
2	2	4	(R1,4,5)
3	3	5	



Realice el ejercicio 10)a) nuevamente considerando este algoritmo, con un quantum de 2 unidades y Timer Variable.

Job	I/O (recur,inst,dur)
1	(R1, 2, 1)
2	(R2, 3, 1) (R2, 5, 2)
4	(R3, 1, 2) (R3, 3, 1)

Job	Llegada	Unidades de CPU
1	0	4
2	2	6
3	3	4
4	6	5
5	8	2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																					
2																					
3																					
4																					
5																					

13.- Suponga que un SO utiliza un algoritmo de VRR con Timer Variable para planificar sus procesos. Para ello, el quantum es representado por un contador, que se decrementa en una unidad cada vez que ocurre una interrupción de reloj.
Bajo este esquema, ¿Puede suceder que el quantum de un proceso nunca llegue a 0 (cero)? Justifique su respuesta.

Si, podría pasar. Supongase que se está trabajando con un *quantum* de cuatro y tenemos un trabajo uno que utiliza el procesador por dos unidades de tiempo, para luego bloquearse por una operación de E/S. EL proceso realiza su operación de E/S, y como tiene prioridad, retoma el procesador, pero como el RR es de timer variable, en lugar de restarle dos unidades de tiempo, le quedan nuevamente cuatro. Suponiendo que el proceso tuviese siempre ráfagas de CPU menores al tamaño del quantum, su contador podría no llegar nunca a cero.

14.- El algoritmo SJF (y SRTF) tiene como problema su implementación, dada la dificultad de conocer la duración de la próxima ráfaga de CPU. Es posible realizar una estimación de la próxima, utilizando la media de las ráfagas de CPU para cada proceso.
Así, por ejemplo, podemos tener la siguiente formula:

$$S_{n+1} = \frac{1}{n} T_n + \frac{n-1}{n} S_n \quad \text{(F1)}$$

Donde:
Ti = duración de la ráfaga de CPU i-ésima del proceso.
Si = valor estimado para el i-ésimo caso
S1 = valor estimado para la primer ráfaga de CPU. No es calculado.

Suponga un proceso cuyas ráfagas de CPU reales tienen como duración: 6, 4, 6, 4, 13, 13, 13 Calcule que valores se obtendrían como estimación para las ráfagas de CPU del proceso si se utiliza la formula (F1), con un valor inicial estimado de S1=10.
S8 = 1/7x 13 + (7-1)/7 x 10
S8= 10,3

La formula anterior (F1) le da el mismo peso a todos los casos (siempre calcula la media). Es posible reescribir la formula permitiendo darle un peso mayor a los casos más recientes y menor a casos viejos (o viceversa). Se plantea la siguiente formula:

$$S_{n+1} = \alpha.T_n + (1-\alpha)S_n \quad \text{(F2)}$$

con 0 < _ < 1
b) Analice para que valores de _ se tienen en cuenta los casos mas recientes.
c) Para la situación planteada en a) calcule que valores se obtendrían si se utiliza la formula (F2) con _=0,2; _=0,5 y _=0,8.
d) Para todas las estimaciones realizadas en a) y c) ¿Cuál es la que mas se asemeja a las ráfagas de CPU reales del proceso?

15.- Colas Multinivel

Hoy en día los algoritmos de planificación vistos se han ido combinando para formar algoritmos más eficientes. Así surge el algoritmo de Colas Multinivel, donde la cola de procesos listos es divida en varias colas, teniendo cada una su propio algoritmo de planificación.
a) Suponga que se tienen 2 tipos de procesos: Interactivos y Batch. Cada uno de estos procesos se coloca en una cola según su tipo. ¿Qué algoritmo de los vistos utilizaría para administrar cada una de estas colas?
Para la cola de procesos interactivos, utilizaría probablemente un Round Robin Virtual, ya que los procesos interactivos suelen tener mayor carga de Entrada/Salida que los procesos Batch, que están ligados a la CPU. En el caso de la cola de Batch, utilizaría algún algoritmo no apropiativo, como First Come First Served o Shortest Job First.

A su vez, se utiliza un algoritmo para administrar cada cola que se crea. Así, por ejemplo, el algoritmo podría determinar mediante prioridades sobre que cola elegir un proceso.
b) Para el caso de las 2 colas vistas en a): ¿Que algoritmo utilizaría para planificarlas?
Utilizaría el algoritmo de prioridades, donde existiera la posibilidad que los procesos de mayor prioridad, los interactivos, pudieran descender en prioridad a lo largo de su vida, para evitar el riesgo de inanición en los procesos Batch.

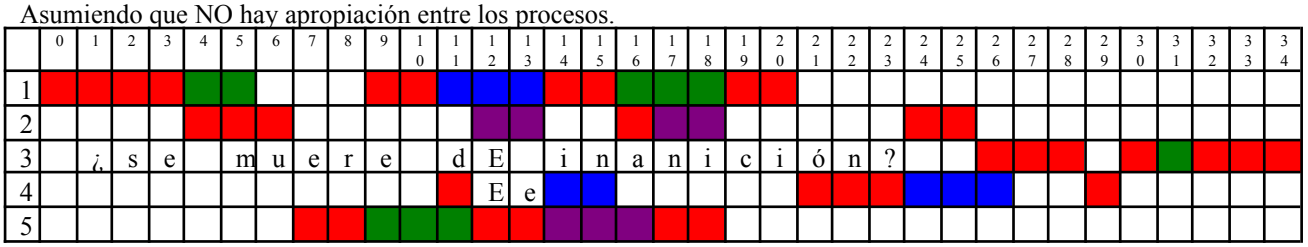
16.- Suponga que en un SO se utiliza un algoritmo de planificación de colas multinivel.

El mismo cuenta con 3 colas de procesos listos, en las que los procesos se encolan en una u otra según su prioridad. Hay 3 prioridades (1, 2, 3), donde un menor número indica mayor prioridad.
Se utiliza el algoritmo de prioridades para la administración entre las colas.
Si se tiene el siguiente lote de procesos a ser procesados con sus respectivas operaciones de I/O.

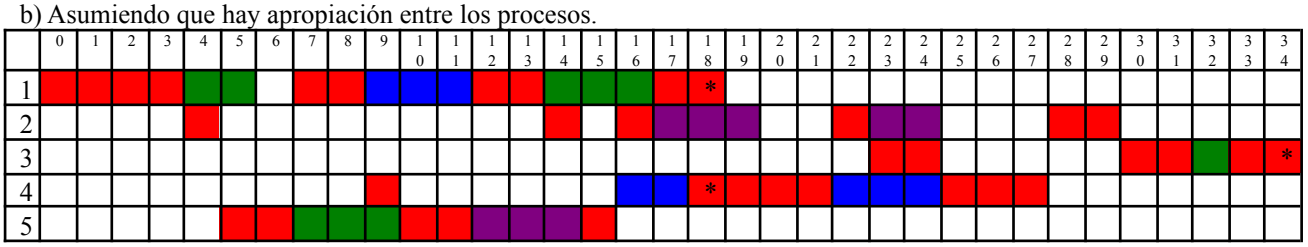
JOB	Inst. Llegada	CPU	I/O (recur,inst,dur)	Prioridad
1	0	9	(R1, 4, 2) (R2, 6, 3) (R1, 8, 3)	1
2	1	5	(R3, 3, 2) (R3, 4, 2)	2
3	2	5	(R1, 4, 1)	3
4	3	7	(R2, 1, 2) (R2, 5, 3)	2
5	5	5	(R1, 2, 3) (R3, 4, 3)	1

Suponiendo que las colas de cada dispositivo se administran a través de FCFS y que cada cola de procesos listos se administra por medio de un algoritmo RR con un quantum de 3 unidades y Timer Variable, realice un diagrama de Gantt:

Asumiendo que NO hay apropiación entre los procesos.



b) Asumiendo que hay apropiación entre los procesos.

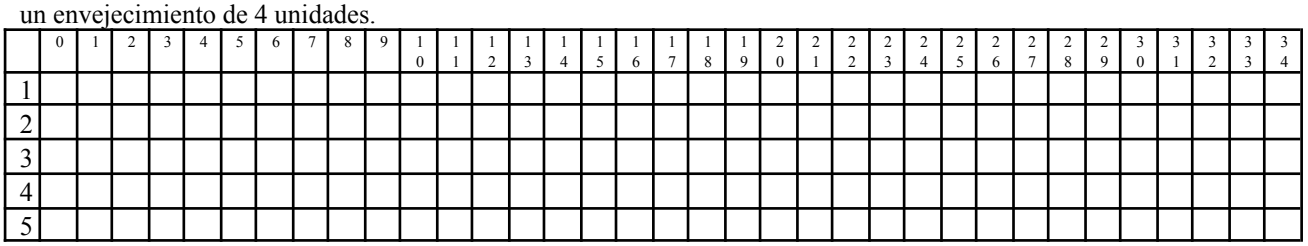


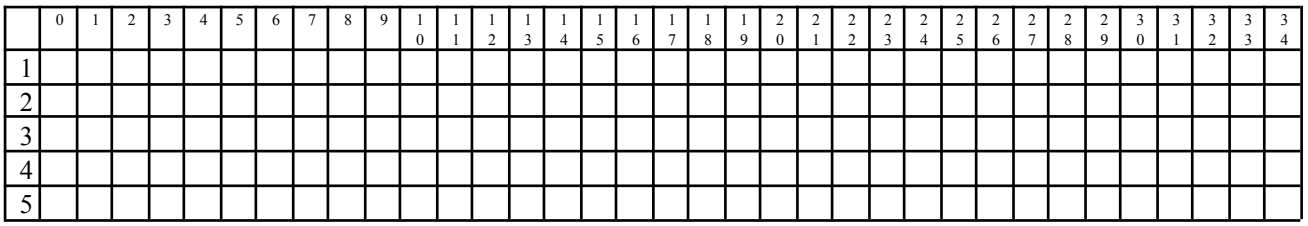
17.- En el esquema de Colas Multinivel, cuando se utiliza un algoritmo de prioridades para administrar las diferentes colas los procesos pueden sufrir starvation.

La técnica de envejecimiento se puede aplicar a este esquema, haciendo que un proceso cambie de una cola de menor prioridad a una de mayor prioridad, después de cierto periodo de tiempo que el mismo se encuentra esperando en su cola. Luego de llegar a una cola en la que el proceso llega a ser atendido, el mismo retorna a su cola original.

Por ejemplo: Un proceso con prioridad 3 esta en cola su cola correspondiente. Luego de X unidades de tiempo, el proceso se mueve a la cola de prioridad 2. Si en esta cola es atendido, retorna a su cola original, en caso contrario luego de sucederse otras X unidades de tiempo el proceso de mueve a la cola de prioridad 1. Esta última acción se repite hasta que el proceso obtiene la CPU, situación que hace que el mismo vuelva a su cola original.

a) Para los casos a) y b) del ejercicio 16) realice el diagrama de Gantt considerando además que se tiene un envejecimiento de 4 unidades.





18.- La situación planteada en el ejercicio 17), donde un proceso puede cambiar de una cola a otra, se la conoce como Colas Multinivel con Realimentación.

Suponga que se quiere implementar un algoritmo de planificación que tenga en cuenta el tiempo de ejecución consumido por el proceso, penalizando a los que más tiempo de ejecución tienen. (Similar a la tarea del algoritmo SJF que tiene en cuenta el tiempo de ejecución que resta).

Utilizando los conceptos vistos de Colas Multinivel con Realimentación indique que colas implementaría, que algoritmo usaría para cada una de ellas así como para la administración de las colas entre sí.

Utilizaría un RR con Q=2 y una retroalimentación de 4.

Tenga en cuenta que los procesos no deben sufrir inanición.

Ok

19.- Un caso real: "*Unix Clasico* " (SVR3 y BSD 4.3)

Estos sistemas estaban dirigidos principalmente a entornos interactivos de tiempo compartido. El algoritmo de planificación estaba diseñado para ofrecer buen tiempo de respuesta a usuarios interactivos y asegurar que los trabajos de menor prioridad (en segundo plano) no sufrieran inanición.
La planificación tradicional usaba el concepto de colas multinivel con realimentación, utilizando RR para cada uno de las colas y realizando el cambio de proceso cada un segundo (quantum). La prioridad de cada proceso se calcula en función de la clase de proceso y de su historial de ejecución. Para ello se aplican las siguientes funciones:

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + nice_j$$

donde:
CPUj(i) = Media de la utilización de la CPU del proceso j en el intervalo i.
Pj(i) = Prioridad del proceso j al principio del intervalo i (los valores inferiores indican prioridad mas alta).
Basej = Prioridad base del proceso j
Nicej = Factor de ajuste
La prioridad del proceso se calcula cada segundo y se toma una nueva decisión de planificación. El propósito de la prioridad base es dividir los procesos en bandas fijas de prioridad. Los valores de CPU y nice están restringidos para impedir que un proceso salga de la banda que tiene asignada. Las bandas definidas, en orden decreciente de prioridad, son:
Intercambio
Control de Dispositivos de I/O por bloques
Gestión de archivos
Control de Dispositivos de I/O de caracteres
Procesos de usuarios.
Veamos un ejemplo:
Supongamos 3 procesos creados en el mismo instante y con prioridad base 60 y un valor nice de 0. El reloj interrumpe al sistema 60 veces por segundo e incrementa un contador para el proceso en ejecución. Los sectores en celeste representan el proceso en ejecución.

Time	Process A		Process B		Process C	
	Priority	CPU Count	Priority	CPU Count	Priority	CPU Count
0	60	0 1 2 • • 60	60	0	60	0
1	75	30	60	0 1 2 • • 60	60	0
2	67	15	75	30	60	0 1 2 • • 60
3	63	7 8 9 • • 67	67	15	75	30
4	76	33	63	7 8 9 • • 67	67	15
5	68	16	76	33	63	7

a) Analizando la jerarquía descripta para las bandas de prioridades: ¿Que tipo de actividad considera que tendrá más prioridad? ¿Por qué piensa que el scheduler prioriza estas actividades?

Tendrán más prioridad las actividades con carga de E/S. El scheduler prioriza estas actividades porque los procesos interactivos tienden a tener mayor carga de entrada salida que de CPU.

b) Para el caso de los procesos de usuarios, y analizando las funciones antes descriptas: ¿Qué tipo de procesos se encarga de penalizar? (o equivalentemente se favorecen). Justifique
Se tiende a penalizar los procesos con mucha carga de CPU y, en contrapartida, se favorecen los procesos con carga de E/S.

c) La utilización de RR dentro de cada cola: ¿Verdaderamente favorece al sistema de Tiempo Compartido? Justifique.
Puede no favorecer, porque Round Robin en general tiende a asignar el tiempo de procesador desigualmente, dando más tiempo a los procesos con mayor carga de CPU, razón por la cual se suele utilizar el Round Robin virtual.
Aparte, si las prioridades en las colas no están correctas, entonces RR funcionará de manera inadecuada.

20.- A cuáles de los siguientes tipos de trabajos:

a) cortos acotados por CPU
b) cortos acotados por E/S
c) largos acotados por CPU
d) largos acotados por E/S
benefician las siguientes estrategias de administración:
a) prioridad determinada estáticamente con el método del más corto primero (SJF).
Cortos acotados por CPU
b) prioridad dinámica inversamente proporcional al tiempo transcurrido desde la última operación de E/S.
largos acotados por E/S

21.- Explicar porqué si el quantum "q" en Round-Robin se incrementa sin límite, el método se aproxima a FIFO

Porque el tiempo de ejecución de un proceso sería igual o mayor al tiempo que el proceso necesita para terminar, por lo tanto se necesitaría que termine o que algo lo bloquee para que otro proceso comience.

22.-Los sistemas multiprocesador pueden clasificarse en:

• Homogéneos: Los procesadores son iguales. Ningún procesador tiene ventaja física sobre el resto.
• Heterogéneos: Cada procesador tiene su propia cola y algoritmo de planificación.
Otra clasificación posible puede ser:
• Multiprocesador débilmente acoplado: Cada procesador tiene su propia memoria principal y canales
• Procesadores especializados: Existe uno o más procesadores principales de propósito general y varios especializados controlados por el primero (ejemplo procesadores de E/S, procesadores Java, procesadores Criptográficos, etc.).
• Multiprocesador fuertemente acoplado: Consta de un conjunto de procesadores que comparten una memoria principal y se encuentran bajo el control de un Sistema Operativo

a) ¿Con cual/es de estas clasificaciones asocia a las PCs de escritorio habituales?
Con el último

b) ¿Que significa que la asignación de procesos se realice de manera simétrica?
En una arquitectura simétrica, el sistema operativo puede ejecutarse en cualquier procesador y cada procesador se autoplanifica con el conjunto de procesos disponibles. Esta solución complica el sistema operativo. El sistema operativo debe asegurarse de que dos procesadores no seleccionan el mismo proceso y que los procesos no se pierden de la cola. Las técnicas deben emplearse para resolver y sincronizar las solicitudes concurrentes de recursos.

c) ¿Qué significa que se trabaje bajo un esquema Maestro/esclavo?
En la arquitectura maestro/esclavo, las funciones clave del núcleo del sistema operativo siempre se ejecutan en un determinado procesador. Los otros procesadores pueden ejecutar sólo programas de usuario. El maestro es el responsable de la planificación de trabajos. Una vez que un proceso está activo, si el esclavo necesita un servicio (por ejemplo, una llamada de E/S), debe enviar una solicitud al maestro

y esperar a que el servicio se lleve a cabo. Esta solución es bastante simple y exige una pequeña mejora en el sistema operativo multiprogramado del monoprocesador. La resolución de conflictos se simplifica puesto que un procesador tiene el control de toda la memoria y todos los recursos de E/S. Las desventajas de esta solución son dos: (1) un fallo en el maestro hace caer todo el sistema y (2), el maestro puede llegar a ser un cuello de botella del rendimiento.

23.-Asumiendo el caso de procesadores homogéneos

a) ¿Cuál sería el método de planificación más sencillo para asignar CPUs a los procesos?
FCFS siempre es el más sencillo.

b) Cite ventajas y desventajas del método escogido
Las ventajas que es un método fácil de implementar y que siempre va a asignarle CPU a los procesos, ya que en un procesador homogéneo no hay prioridades.
La desventaja es que los tiempos de retorno son altos y si un proceso se bloquea, tendrá que esperar mucho tiempo para terminar, aunque sea importante.

23.-Indique brevemente a que hacen referencia los siguientes conceptos:

a) Huella de un proceso en un procesador c) ¿Por qué podría ser mejor en algunos casos que un proceso se ejecute en el mismo procesador? f) Compare los conceptos de afinidad y balanceo de carga y como uno afecta al otro.
A medida que proceso se ejecuta en un procesador determinado va dejando una huella (estado del proceso en la memoria cache del procesador donde se ejecuta), por lo que parece adecuado mantener a un proceso en el mismo procesador (criterio de afinidad al procesador). Sin embargo, las políticas que potencian la afinidad al procesador tienden a provocar un desequilibrio en la carga de los procesadores, que se traduce en una utilización menos eficiente de los recursos. El criterio opuesto (reparto de la carga), provoca sobrecargas en el bus de memoria al invalidar frecuentemente el contenido de las memorias caché.

b) Afinidad con un procesador
La afinidad que tiene el hilo con los procesadores es el conjunto de procesadores de un sistema multiprocesador que pueden ejecutar al hilo; este conjunto es igual o es un subconjunto de la afinidad que tiene el proceso con los procesadores.

d) ¿Puede el usuario en Windows cambiar la afinidad de un proceso? ¿Y en linux?
En windows no es posible, pero en linux se puede hacer a través de la instrucción taskset
taskset <affinity mask> -p <process>
La máscara de bits puede ser una lista (es decir, 1,3,4 utilizar núcleos de 1 3 y 4 de un sistema central 4 +) o una máscara de bits en hexadecimal (0x0000000D el 1,3,4, 0x00000001 básico por sólo 1)

e) Investigue el concepto de balanceo de carga (load balancing).
El balance o balanceo de carga es un concepto usado en informática que se refiere a la técnica usada para compartir el trabajo a realizar entre varios procesos, ordenadores, discos u otros recursos. Está íntimamente ligado a los sistemas de multiprocesamiento, o que hacen uso de más de una unidad de procesamiento para realizar labores útiles.
El balance de carga se mantiene gracias a un algoritmo que divide de la manera más equitativa posible el trabajo, para evitar los así denominados cuellos de botella.

24.-Si a la tabla del ejercicio 6 la modificamos de la siguiente manera:

Job	Llegada	Unidades de CPU	Procesador preferido
1	0	4	CPU0
2	2	6	CPU0
3	3	4	CPU1
4	6	5	CPU1
5	8	2	CPU0

Y considerando que el scheduler de los Sistemas Operativos de la familia Windows utiliza un mecanismo denominado preferred processor (procesador preferido). El scheduler usa el procesador preferido a modo

de afinidad cuando el proceso esta en estado ready. De esta manera el scheduler asigna este procesador a la tarea si este está libre.

a) Ejecute el esquema anterior utilizando el algoritmo anterior

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1														C	P	U	0	C	P	U	1
2																					
3								PR	E	G	U	NT	A	R	SI	E	M	PI	E	Z	A
4																					
5																					

Ejecute el esquema anterior. Pero ahora si el procesador preferido no está libre es asignado a otro procesador. Luego el procesador preferido de cada job es el último en el cual ejecuto.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1														C	P	U	0	C	P	U	1
2																					
3																					
4																					
5																					

c) Para cada uno de los casos calcule el tiempo promedio de retorno y el tiempo promedio de espera

A		B	
TR	TE	TR	TE
1:4	1:0	1:4	1:0
2:8	2:2	2:6	2:0
3:4	3:0	3:5	3:1
4:7	4:1	4:7	4:1
5:4	5:2	5:2	5:0
PROM: 5,4	PROM: 1	PROM: 4,8	PROM:0,4

d) ¿Cuál de las dos alternativas planteadas es mas performante?

La alternativa B es la más PERFORMANTE!