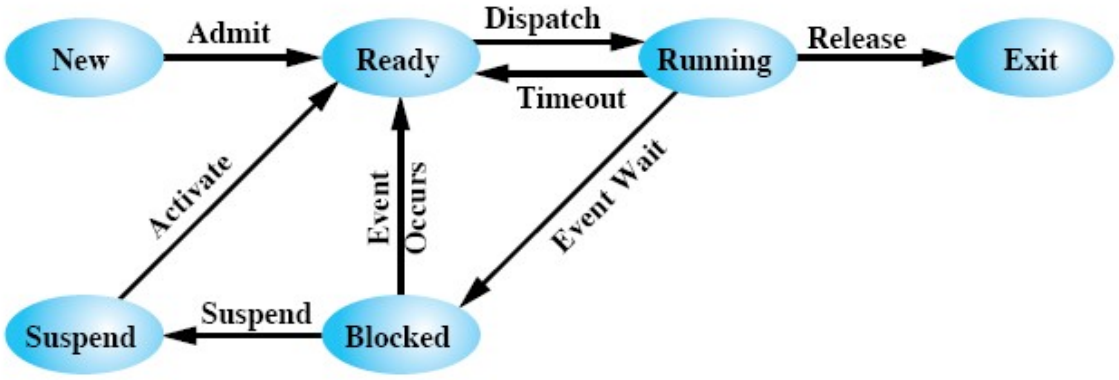
Practica 4 - ISO

1) **a) Proceso:** Programa en ejecución, es dinámico, posee program counter, su ciclo de vida comprende desde que se ejecuta hasta que termina. Según su historial de ejecución se puede clasificar en CPU Bound (ligados a la CPU) o I/O Bound (ligados a entrada / salida).
Programa: Código binario estático, no posee program counter, existe desde que se edita hasta que se borra.
b) TR: El tiempo de retorno para un trabajo (job) es el tiempo que transcurre entre que el proceso llega al sistema hasta que completa su ejecución.
TE: El tiempo que el proceso se encuentra en el sistema esperando, es decir el tiempo que pasa sin ejecutarse (TR - Tcpu).
c) TPR: Es el tiempo promedio entre todos los tiempos de retorno.
TPE: Es el tiempo promedio de todos los tiempos de espera.
d) El Quantum es el ciclo de reloj que determina cuanto tiempo podrá usar el procesador cada proceso.
e) Apropiativo (Preemptive): El proceso en ejecución puede ser interrumpido por el SO y llevado a la cola de listos.
No-Apropiativo (Nonpreemptive): Una vez que un proceso esta en estado de ejecución, continua hasta que termina o se bloquea por algún evento (e.j. I/O).
f) Short-Term-Scheduler: Determina que proceso de la cola de listo pasará a ejecutarse.
Medium-Term-Scheduler: Realiza el swapping (intercambio) entre el disco y la memoria cuando el SO lo determina (puede disminuir o aumentar el grado de multiprogramación).
Long-Term-Scheduler: Admite nuevos procesos en estado Nuevo a memoria (controla el grado de multiprogramación).
g) El dispatcher es el modulo encargado de hacer el context switch que indica el short term escheduler, es decir, es el que cambia el contexto de los procesos y ejecuta otro diferente, el que alterna el modo de ejecución, el que salta a la siguiente instrucción de otro proceso diferente (cambia de proceso en ejecucion), etc.

2) **a)** La información mínima que se necesita principalmente es PID, el estado, la prioridad, el PC (Program Counter), punteros a memoria, informacion principal del programa y de E/S, entre otros datos. Dicha información recide en la PCB (Process Control Block), que es una tabla que es lo primero que se crea cuando se crea un proceso y es lo ultimo que se borra.
b) CPU-Bound: Significa que el proceso requiere mucho tiempo de su ejecución en CPU (tienden a estar ligados a el).
I/O-Bound: Son los que están ligados a entrada y Salida.
c) Un proceso puede atravesar por los estados: New (nuevo), Ready (listo), Running (corriendo/ en ejecucion), Blocked (Bloqueado esperando un evento externo), Supend (suspendido en memoria secundaria) y Finished (el proceso termino de ejecutarse).
d)

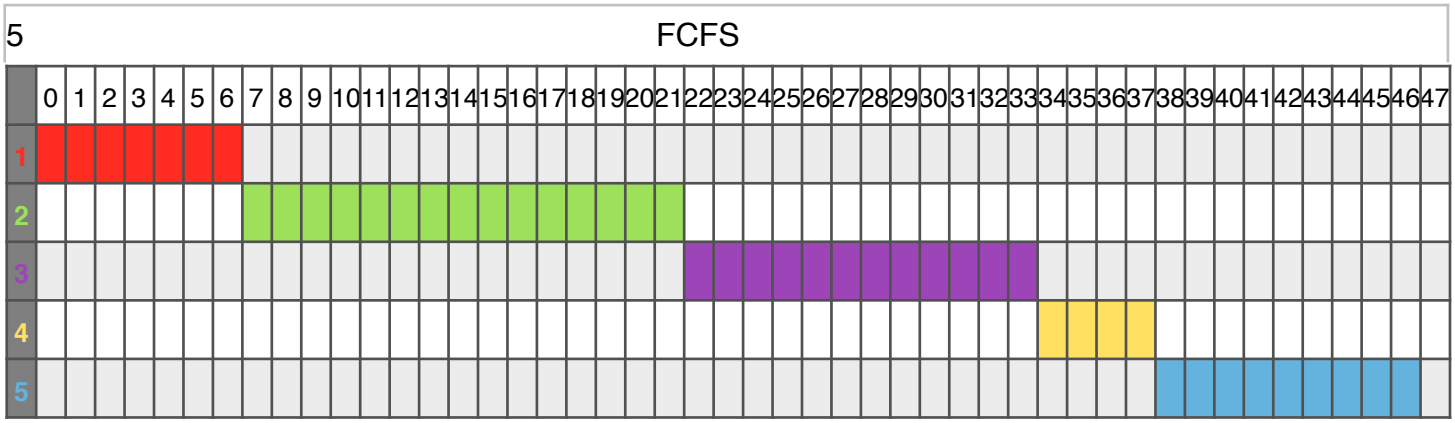


e) El Long Term se encarga del New, el es le encargado de cargar el proceso en memoria para su ejecucion, luego esta el Short Term que se encarga de la transición de Ready a Running y viceversa, y por ultimo el Medium Term es el encargado de hace el Swapping a los procesos bloqueados para dejarlos en memoria secundaria y dar espacio a otros que necesitan estar en la cola para ejecutarse.

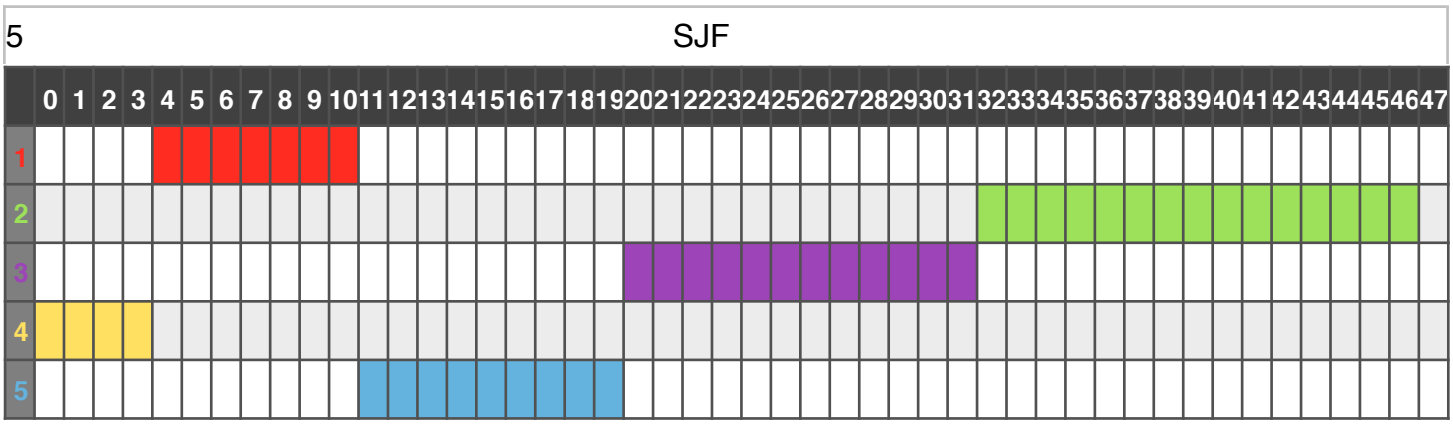
3) a) FCFS (First Come First Served): Este algoritmo es no apropiativo y su criterio de elección para ejecutar procesos es su orden de llegada: el primero que viene, es el primero que se ejecuta.
SJF (Short Job First): No apropiativo, ejecuta el proceso mas corto primero.
Round Robin: Es apropiativo, establece un Quantum, cuando este termina se quita el proceso que se esta ejecutando de la CPU y se lo manda a la cola (en caso de que todavía no se haya terminado de ejecutar obviamente) para seguir ejecutándose luego.
Prioridades: Es apropiativo o no, le establece un valor de prioridad a cada proceso, cuanto menor es ese valor, mayor prioridad tiene ese proceso.
b) Si, el de Round Robin, que necesita saber el Quantum, así como si se trata de TV o TF para darse cuenta cuando debe de cambiar de proceso. Los datos que precisan FCFS y el de prioridades no serian parámetros del algoritmo.
c) El FCFS no beneficia a ningún proceso en particular, el de prioridades beneficiaria a los de E/S, SJF beneficia a los de E/S ya que estos utilizan ráfagas cortas de CPU y realizan E/S. Los algoritmos RR intentan ser equitativos aunque terminan perjudicando a los de E/S que no completan sus ráfagas con el quantum asignado debido a que deben la CPU por E/S.
d) Es lo mismo que el 3-C.

4) a) Timer Variable: El contador(Quantum) se inicializa en un valor Q cada vez que un proceso es asignado a la CPU.
Time Fijo: El contador toma ese valor Q cuando llega a 0, y asi cíclicamente. Es como si el contador se compartiera entre los procesos.
b) Si un proceso tiene asignado un quantum de 4, y luego de realizar 2 quantum solicita el uso de E/S en timer variable, el nuevo proceso tendría asignado nuevamente 4 quantum, mientras que en timer fijo solo se asignarían 2.
c) En el variable el Quantum debería almacenarse uno por cada proceso, en cambio en el Timer fijo podría ser una variable global accesible por todos.

5)

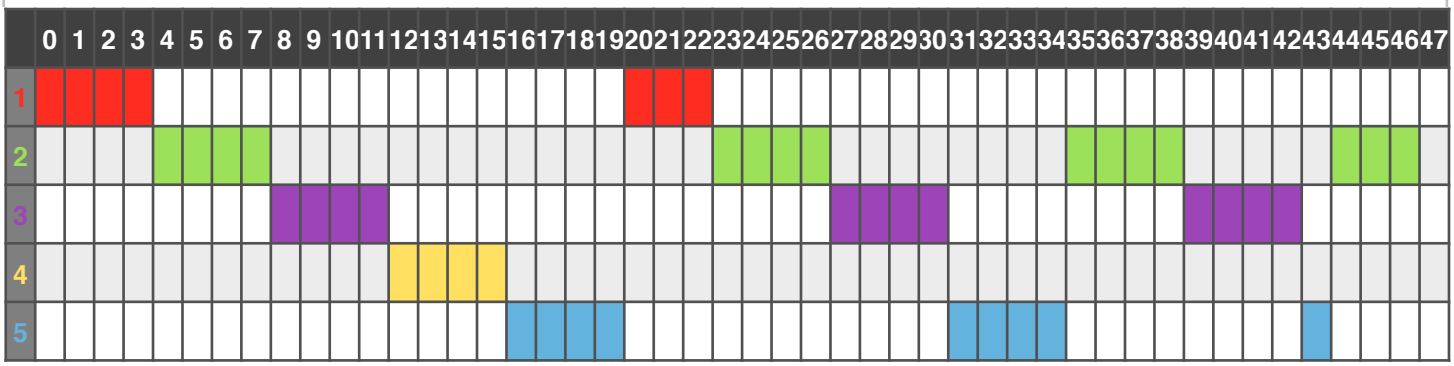


1)TE: 0 TR: 7 2)TE: 7 TR: 22 3)TE: 22 TR: 34 4)TE: 34 TR: 38 5)TE: 38 TR: 47 **TPE:** 20.2 **TPR:** 29.6



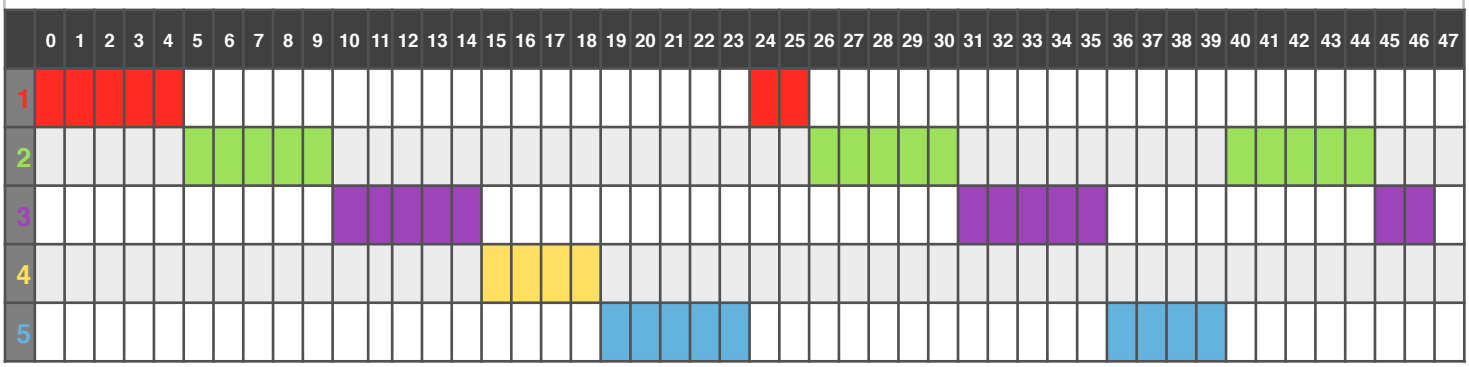
1)TE: 4 TR: 11 2)TE: 32 TR: 47 3)TE: 20 TR: 32 4)TE: 0 TR: 4 5)TE: 11 TR: 20 **TPE:** 13.7 **TPR:** 22.8

5 Round Robin Quantum: 4 Timer Variable (FCFS)



1)TE: 16 TR: 23 2)TE: 32 TR: 47 3)TE: 28 TR: 40 4)TE: 12 TR: 16 5)TE: 32 TR: 41**TPE:** 24 **TPR:** 33.4

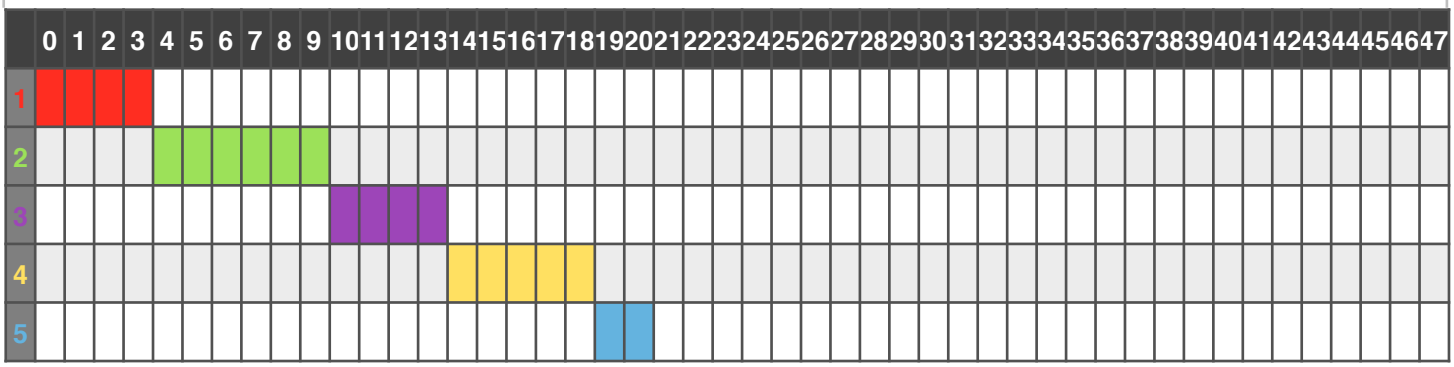
5 Round Robin Quantum: 5 Timer Fijo (FCFS)



1)TE: 19 TR: 26 2)TE: 30 TR: 45 3)TE: 35 TR: 47 4)TE: 15 TR: 19 5)TE: 31 TR: 40**TPE:** 26 **TPR:** 35.4

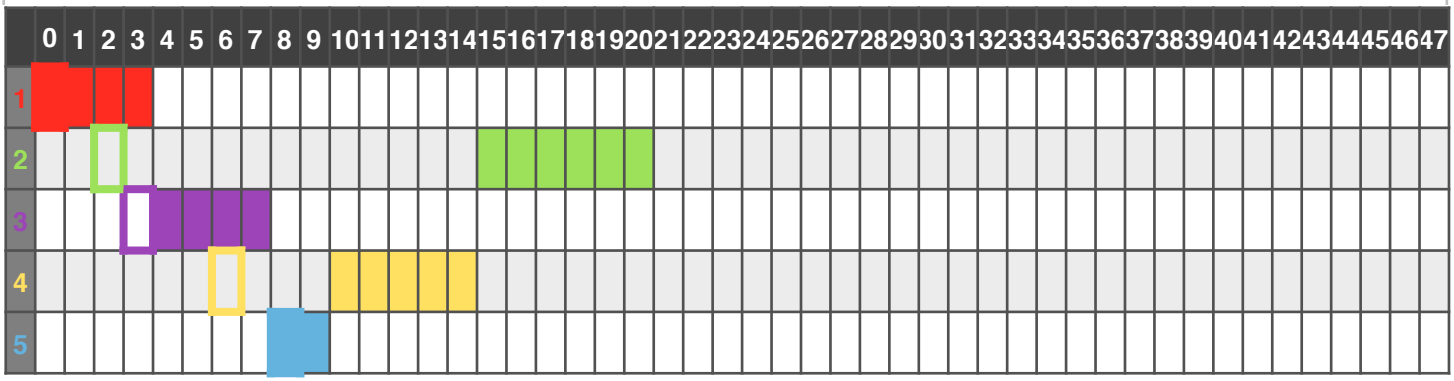
6)

6 FCFS



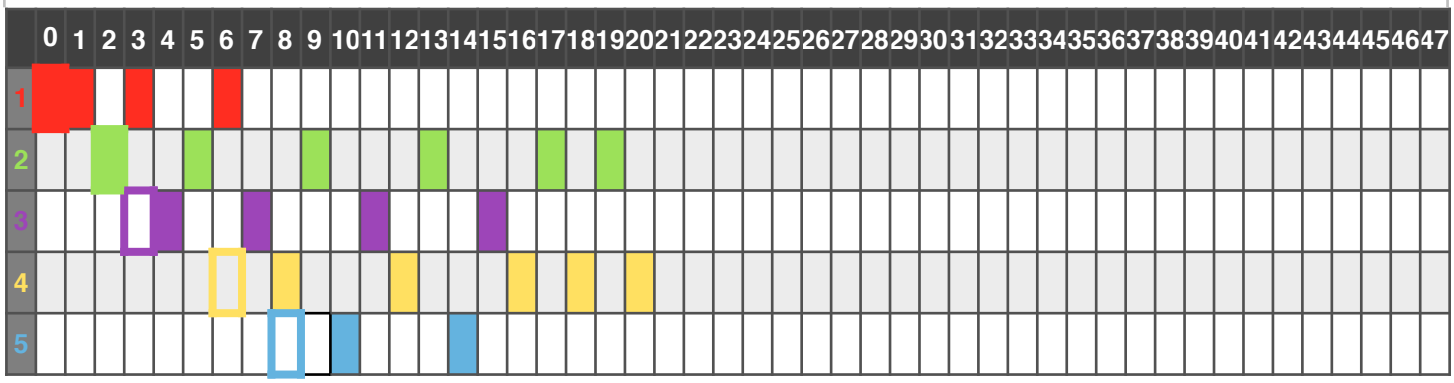
1)TE: 0 TR: 4 2)TE: 2 TR: 8 3)TE: 7 TR: 11 4)TE: 8 TR: 13 5)TE: 11 TR: 13 **TPE:** 5.6 **TPR:** 9.8

Tabla 1-1-2-1-1-1



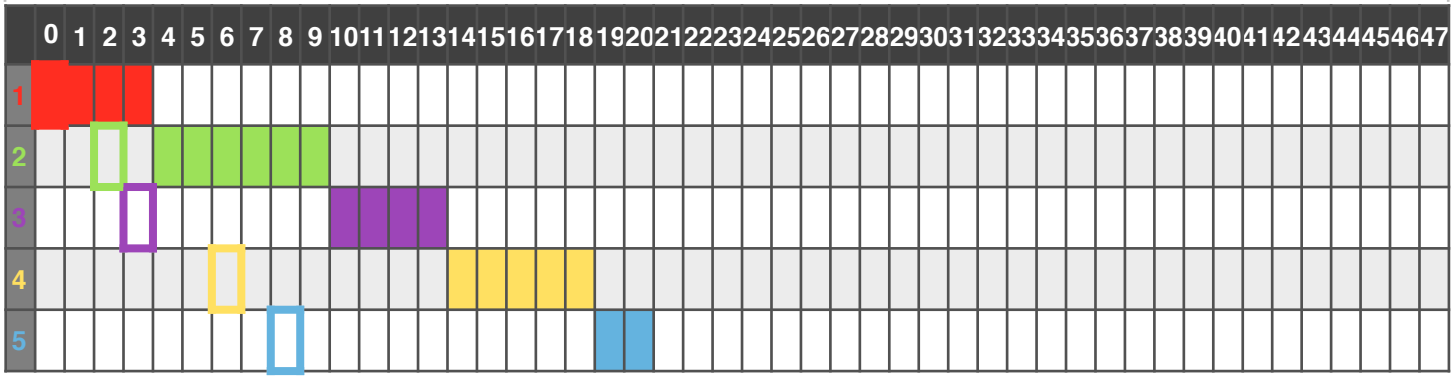
1)TE: 0 TR: 4 2)TE: 13 TR: 19 3)TE: 1 TR: 5 4)TE: 4 TR: 9 5)TE: 0 TR: 2 **TPE:** 3.6 **TPR:** 7.8

Round Robin Quantum: 1 Timer Variable (FCFS) (El mas nuevo)



1)TE: 3 TR: 7 2)TE: 12 TR: 18 3)TE: 9 TR: 13 4)TE: 10 TR: 15 5)TE: 5 TR: 7 **TPE:** 7.8 **TPR:** 12

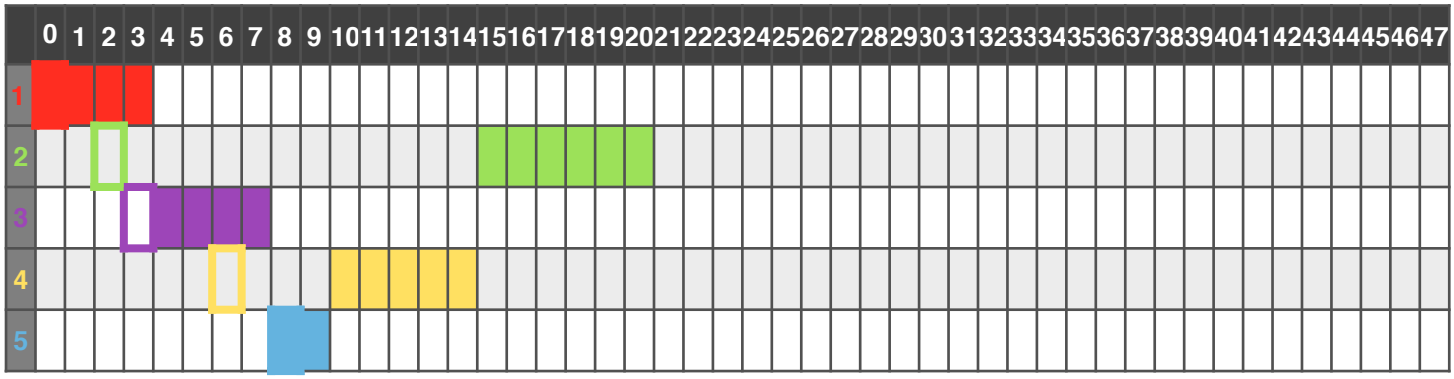
Round Robin Quantum: 6 Timer Variable (FCFS) (El mas nuevo)



1)TE: 0 TR: 4 2)TE: 2 TR: 8 3)TE: 7 TR: 11 4)TE: 8 TR: 13 5)TE: 11 TR: 13 **TPE:** 5.6 **TPR:** 9.8

7) a)

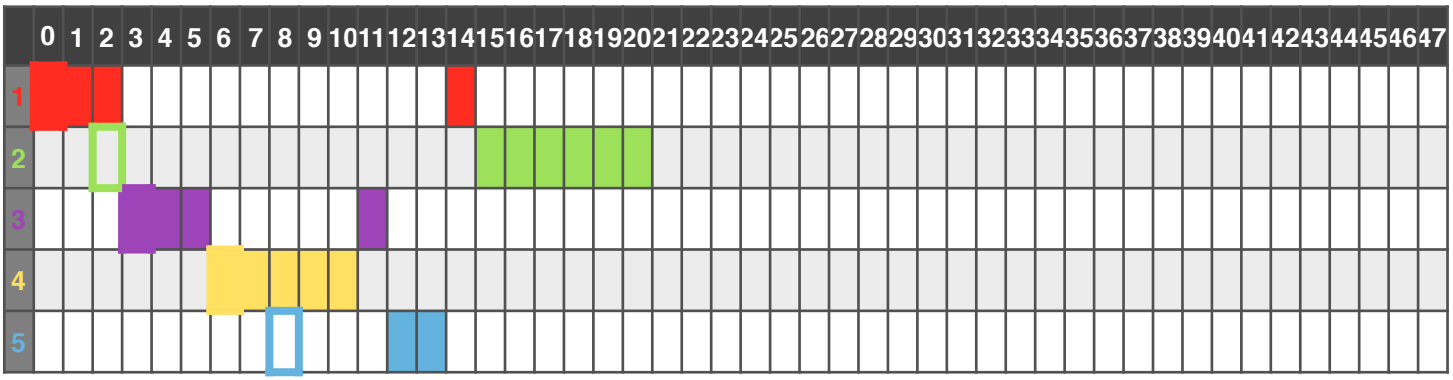
SRTF



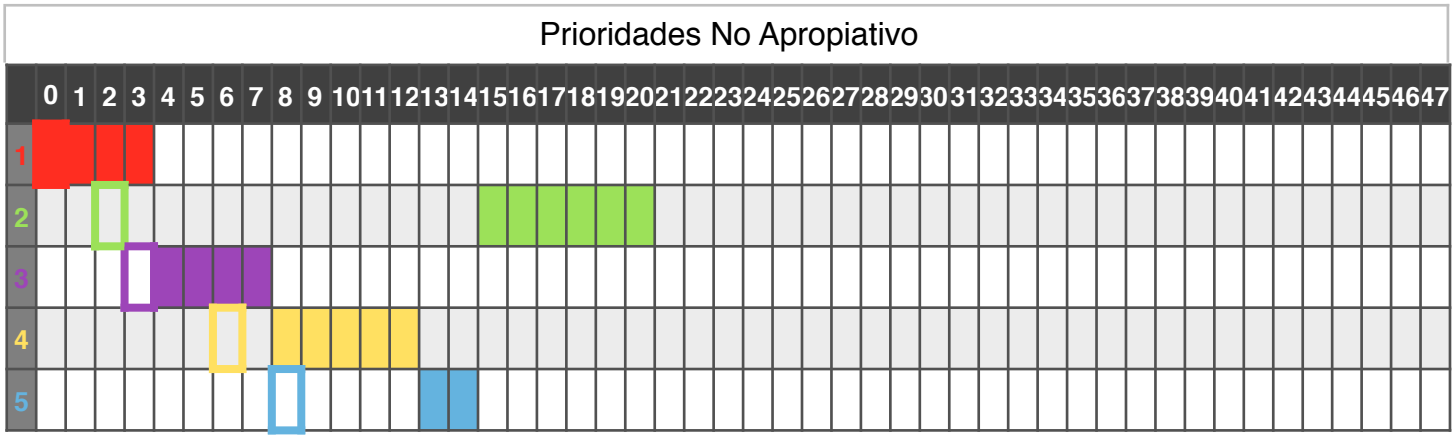
b) Tiene mayor ventaja con respecto a los algoritmos SJF debido a que se atienden mas rápido los procesos cortos.

8)

Prioridades Apropiativo (FCFS)



1)TE: 11 TR: 15 2)TE: 13 TR: 19 3)TE: 5 TR: 9 4)TE: 0 TR: 5 5)TE: 4 TR: 6 **TPE:** 6.6 **TPR:** 10.8



1)TE: 11 TR: 15 2)TE: 13 TR: 19 3)TE: 5 TR: 9 4)TE: 0 TR: 5 5)TE: 4 TR: 6 **TPE:** 6.6 **TPR:** 10.8

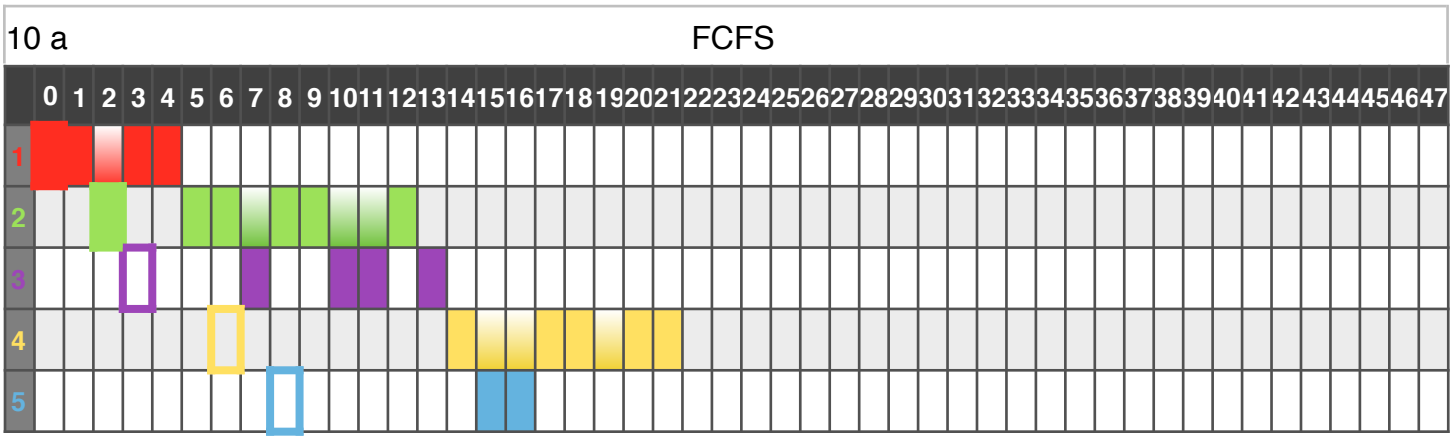
c) Estos Algoritmos de Prioridades los utilizaría bajo circundadas donde se necesite procesos de tiempo corto (E/S), debido a que se los atiende mas rápido en cazo de tener importancia al momento de ejecutarse.

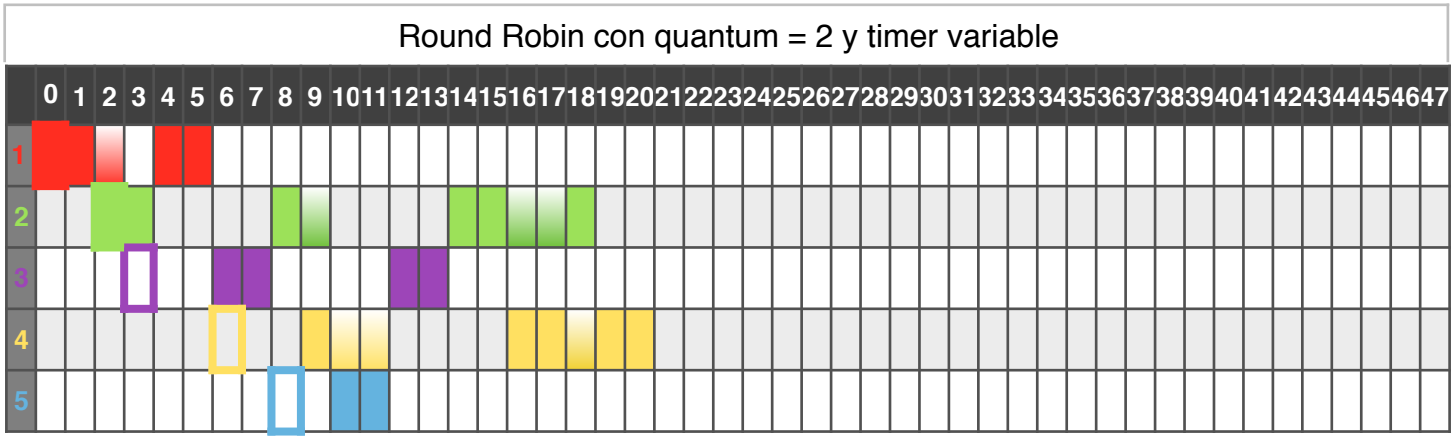
9) a) Inanición / starving: es un problema relacionado con los sistemas multitarea, donde a un proceso o un hilo de ejecución se le deniega siempre el acceso a un recurso (CPU, impresora, etc) compartido. Sin este recurso, la tarea a ejecutar no puede ser nunca finalizada.

b) Lo podría provocar el SJF, el SRJF, o el de prioridades apropiativas, ya que si llegaran muchos procesos cortos o de mayor prioridad un proceso largo o con baja prioridad no podría ejecutarse nunca.

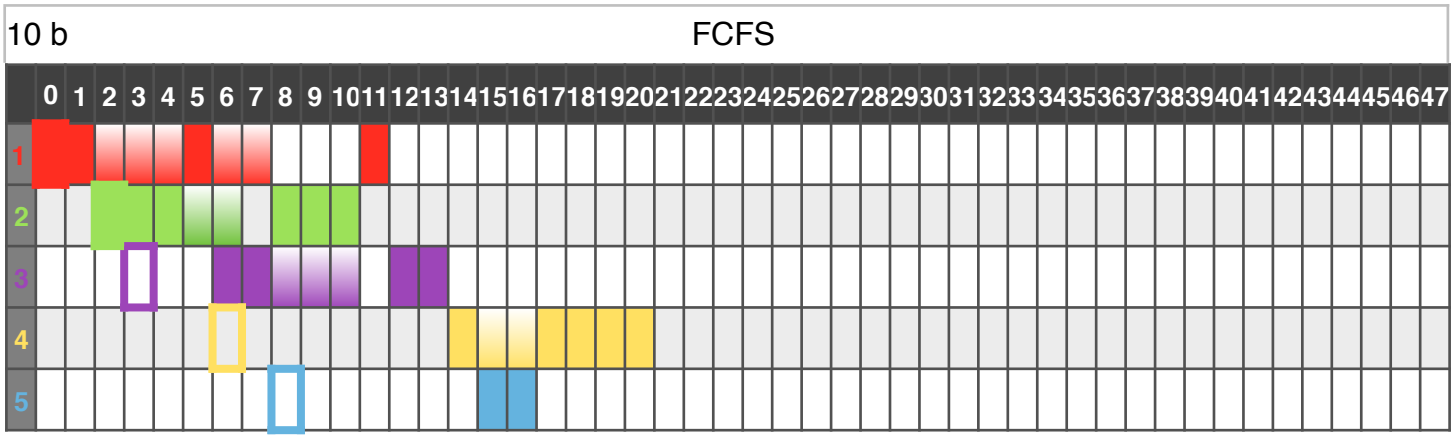
c) Si (Burbujeo), en el caso de las prioridades si un procesos no se ejecuto en un lapso de tiempo entonces se le incrementa la prioridad, de ese modo en algún momento tendrá la prioridad suficiente como para ejecutarse.

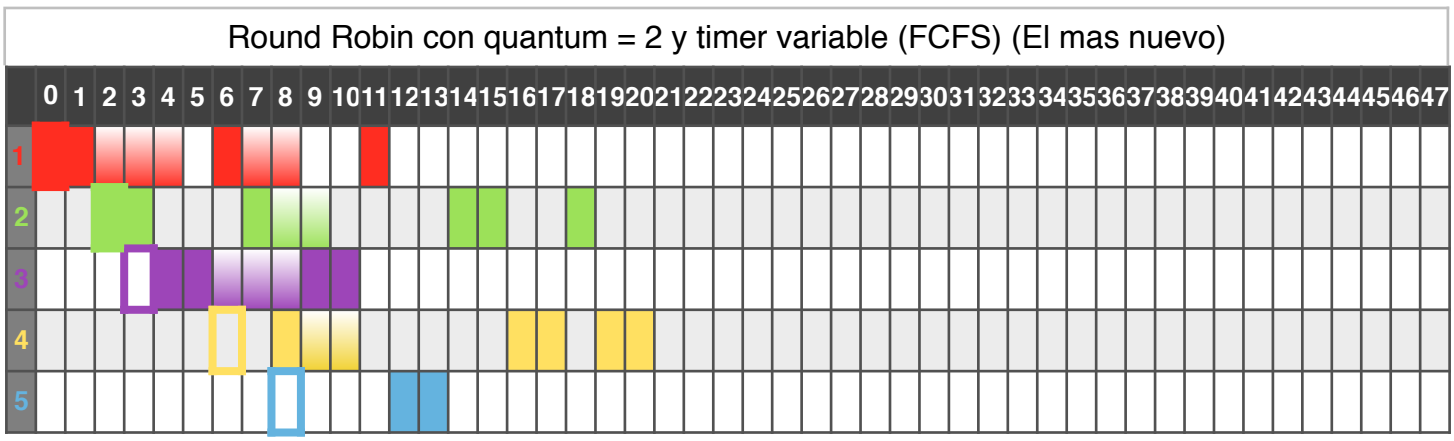
10) a)





b)





11) **a)** Los algoritmos Rond Robin tienen ciertas desventajas con los procesos ligados a E/S debido a que tarda mas en ejecutarse en dicho caso.
b) Los algoritmos SRTF tienen desventajas con procesos ligados a CPU debido a que los procesos de E/S tienen tiempo de ejecución corto y se ejecutarían primero.

12) Round Robin no seria lo optimo para E/S, ya que no suele completar su quantum. El SRTF beneficia E/S , ya que el proceso mas corto es el cual se ejecuta primero y como los procesos de E/S son los que tienen menor tiempo de CPU se ejecutan primero.

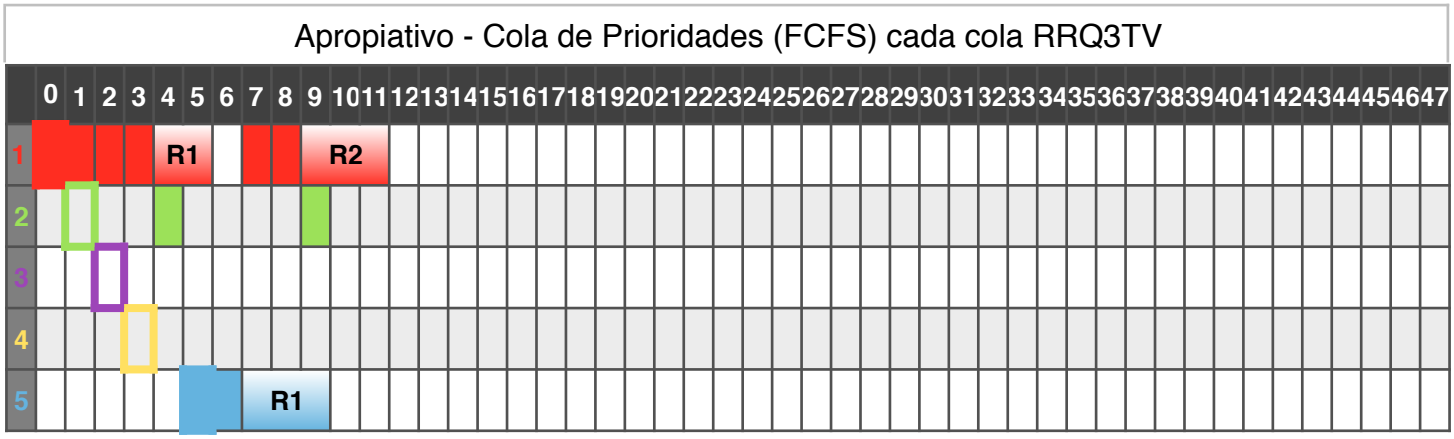
13) Si, puede llegar a ocurrir en el caso de que el proceso necesite muchos menos ciclos de reloj que el que tienen el clock para generar la interrupción.

14) $S_{n+1} = {}^{1}T_n + {}^{n+1}S_n$ 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.
a)

15) a) Para la cola de procesos interactivos usaría el algoritmo Round Robin con Quantum grande con prioridades (no apropiativo); mientras que para los procesos batch utilizaría un algoritmo basado en prioridades, o un SRTF (dependiendo de los procesos batch que tenga).

b) Utilizaria un algoritmo con prioridades, daria mayor prioridad a los procesos E/S bound, ya que son los que interactúan con el usuario y necesitan ser atendidos inmediatamente. Para evitar la inanición tendría que haber un algoritmo que aumente la prioridad de la cola que contiene los procesos batch cada determinado lapso de tiempo sin ejecutarse ninguno de los procesos que la contienen.

16) a)



b)

17)
18)
19)
20) Para: - Cortos acotados por CPU se beneficiaria los de prioridad determinada estáticamente con el método del más corto primero (SJF).
- Cortos acotados por E/S se beneficiaria los de prioridad dinámica inversamente proporcional al tiempo transcurrido desde la última operación de E/S.
- Largos acotados por CPU se beneficiaria los de prioridad dinámica inversamente proporcional al tiempo transcurrido desde la última operación de E/S.
- Largos acotados por E/S se beneficiaria los de prioridad dinámica inversamente proporcional al tiempo transcurrido desde la última operación de E/S.

21) Porque al ser cada vez mas grande si el proceso tiene menos ciclos de uso del CPU entonces va a tener que ejecutarse todo ya que el quantum no lo va a sacar de la CPU, por lo que seria un FCFS.

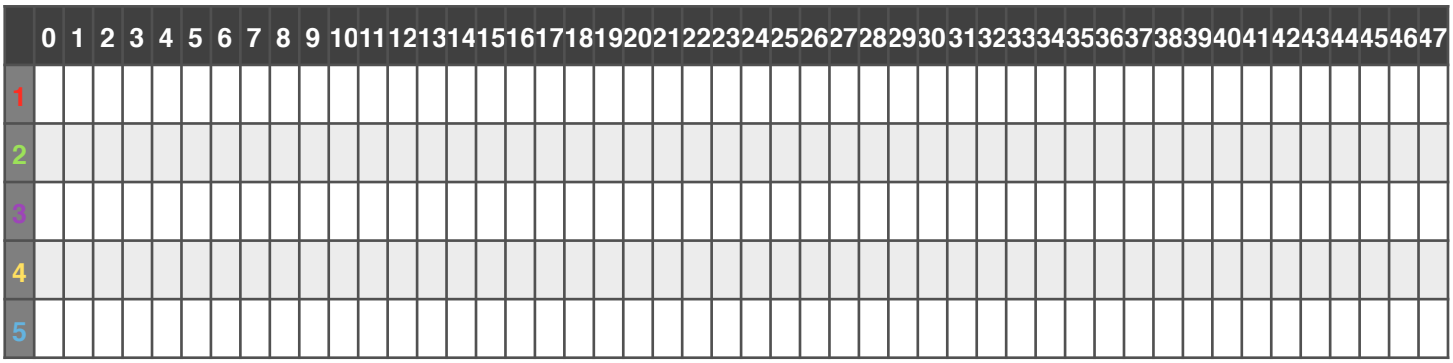
22) a) Asociaría a las PCs de escritorio a una clasificación con procesadores homogéneos, y con un multiprocesador fuertemente acoplado.

b) La asignación simétrica de procesos quiere decir que el sistema operativo puede ejecutarse en cualquier procesador y cada procesador se autoplanifica con el conjunto de procesos disponibles. El sistema operativo debe asegurarse de que dos procesadores no seleccionan el mismo proceso y que los procesos no se pierden de la cola.
c) 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.

23) a) Para procesadores Homogéneos el método de planificación más sencillo para asignar CPUs a dichos procesos seria FCFS (First Come First Served).
b) Las ventajas y desventajas del método FCFS para procesadores homogéneos son
Ventajas: Optimización de tiempo de CPU al momento de elegir el proceso a ejecutarse.
Desventajas: Procesos que requieran realizar E/S provocarían tiempo ocioso del procesador.

24) a) 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).
b) Afinidad al procesador hace referencia a que todo proceso mantenga la huella para que pueda ejecutarse siempre en el mismo procesador manteniendo siempre esa union con el procesador que se ejecuto la primera ves. 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.
c) El criterio de afinidad al procesador seria beneficioso debido a que algunos procesos seria mejor ejecutarlos en procesadores con la mejor planificación para ellos.
d) En Windows y en Linux el usuario puede cambiar la afinidad de procesos al procesador.
e) El equilibrio de carga (Load Balancing) tiene como objetivo optimizar el uso de recursos, maximizar el rendimiento , minimizar el tiempo de respuesta, y evitar la sobrecarga de cualquier recurso individual.
f) El concepto de afinidad de procesos y el equilibrio de carga, son en teoría contrarios, debido a que la afinidad lleva a los procesos a ejecutarse en un determinado procesador sobrecargando dicho procesador, caso contrario el equilibrio de carga que reorganiza los procesos para justamente equilibrar la carga de los procesadores y repartirlos equitativamente en todos los procesadores.

25)



1)TE: TR: 2)TE: TR: 3)TE: TR: 4)TE: TR: 5)TE: TR: **TPE: TPR:**