Sistemas Operativos
1° Parcial 1C2018 - TM - Resolución

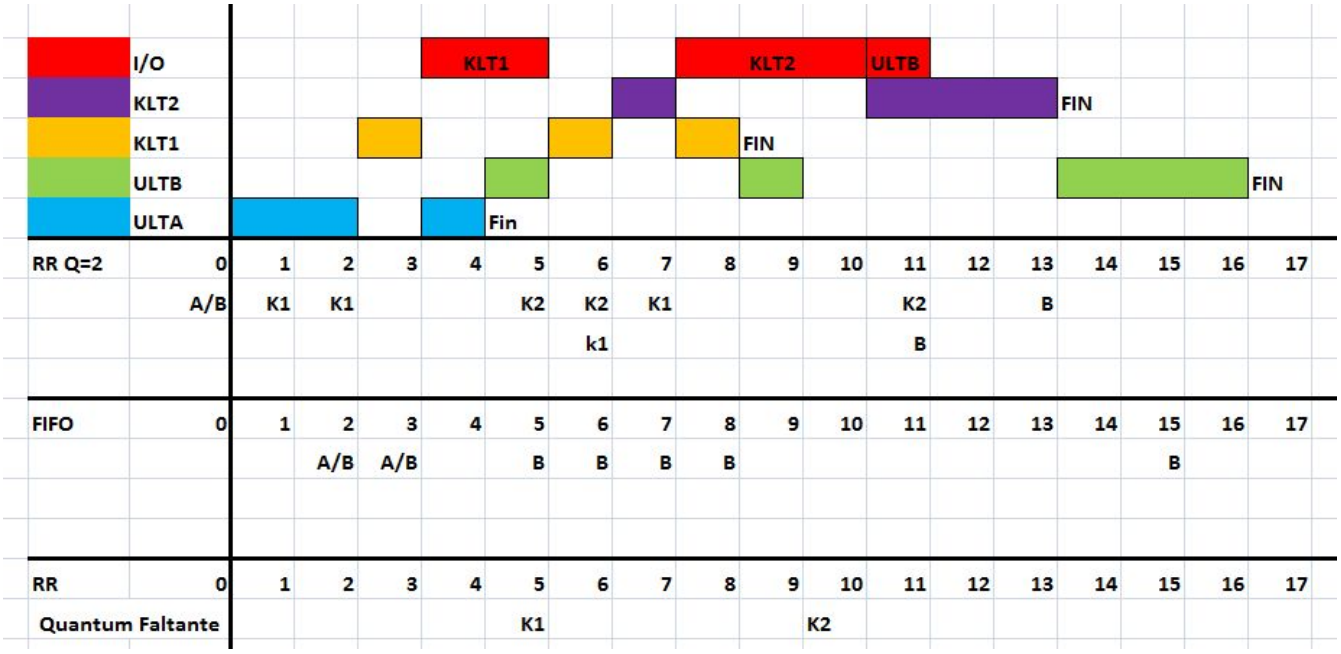
Aclaración: La mayoría de las preguntas o ejercicios no tienen una única solución. Por lo tanto, si una solución particular no es similar a la expuesta aquí, no significa necesariamente que la misma sea incorrecta. Ante cualquier duda, consultar con el/la docente del curso.

Teoría

1. El proceso no es consciente de que se bloquea o que continua su ejecución. Esto se debe a que dichos eventos son realizados por el sistema operativo, guardando y restaurando el contexto de ejecución de forma transparente al proceso.
2.
    a. Retención y espera: el proceso debe solicitar todos los recursos que va a utilizar de forma simultánea, y si alguno de los mismos no está disponible, la totalidad de la solicitud es denegada.
    b. Sin desalojo: el sistema operativo debe poder ex-propiar los recursos que el proceso tenga asignados, para lo cual los mismos deben poder ser retrotraidos a un estado previo consistente, y los procesos deben disponer de un mecanismo para ser notificados que les fué des-asignado un recurso.
3. Los semáforos mutex resuelven el problema de condición de carrera, es decir, que el resultado de la ejecución sea distinta según el orden de ejecución por estar accediendo en forma concurrente (al menos uno en modo escritura) a un recurso compartido. Esto se puede lograr también con instrucciones atómicas como testAndSet o swapAndExchange. Si un semáforo tiene un valor negativo puedo darme cuenta que el wait está implementado con bloqueo, y que hay al menos un proceso bloqueado en la cola de espera de dicho mutex.
4.
    a. Verdadero. Un ejemplo de esto es la inversión de prioridades, en el que un proceso de mayor prioridad no puede ejecutar por estar esperando un recurso retenido por uno de menor prioridad
    b. Verdadero. En el caso de la recuperación, si siempre se elige al mismo proceso para ser finalizado o desalojado, podría ocurrir que nunca termine su ejecución. En el caso de prevención, podría ocurrir lo mismo, que cuando se bloquee esperando un recurso venga otro que necesite alguno de los que retiene y sea desalojado reiteradas veces.
5. A mayor quantum el algoritmo RR degenera en FIFO. A menor quantum se vuelve más equitativo pero con mayor overhead. En el algoritmo VRR afectaría de forma similar, con el agregado de tener una cola auxiliar potencialmente grande con un quantum chico, y una degeneración al algoritmo RR común con un quantum grande.

Practica

1. A)



1b1) En el instante cero que comienza ultb porque tiene una rafaga mas pequeña que ULTA

2.

SA = 1 SL= 1 SB = 0 SR = 0 SLBR =0

Proceso 1	Proceso 2	Proceso 3	Proceso 4
wait(SR) wait(SLBR) **print(R)** signal(SA) signal(SL)	wait(SB) wait(SLBR) **print(B)** signal(SA) signal(SR)	wait(SA) **print(A)** signal(SLBR)	wait(SL) wait(SLBR) **print(L)** signal(SB) signal(SA)

3.

Peticiones actuales

	R1	R2	R3	R4	R5
P1	1	0	1	0	0

Asignaciones

	R1	R2	R3	R4	R5
P1	0	0	0	0	0

P2	2	1	0	0	0
P3	2	0	1	0	0
P4	0	0	0	0	0
P5	1	1	0	0	0
P6	1	0	1	0	0

P2	2	0	1	0	0
P3	0	2	0	1	0
P4	0	0	0	2	0
P5	0	0	0	0	2
P6	1	1	1	0	0

Vector recursos totales = [4, 4,2,3, 4]

a) Procesos que no se encuentran en deadlock:
   i) P1 -> No tiene recursos asignados, no cumple con "retención"
   ii) P4 -> No tiene peticiones pendientes, no cumple con "espera"
   iii) P5 -> se ve que el recurso que está reteniendo no está sido pedido por ninguno, no está dentro de ningún ciclo

b) Aplicamos algoritmo de detección de deadlocks
   Recursos disponibles = 1 1 0 0 2
   Ejecuta P5 -> 1 1 0 0 4
   También podemos asumir que en algún momento P4 va a finalizar ya que no tiene más peticiones -> 1 1 0 2 4
   No ejecuta ninguno más -> deadlock

c) Finalizó al proceso P2
   Recursos disponibles -> 3 1 1 0 2
   P1 -> 3 1 1 0 2
   P3 -> 3 3 1 1 2
   P5 -> 3 3 1 1 4
   P6 -> 4 4 2 1 4
   FInaliza p4 -> 4 4 2 3 4
   Luego podría ejecutar nuevamente desde el inicio P2

d) Basándose en sus grandes conocimientos de sistemas operativos indique simplemente mirando la foto actual cuáles procesos no se encuentran dentro del problema justificando.

e) Se puede agregar una instancia de R3
   Recursos disponibles 1 1 1 0 2
   P6 -> 2 2 2 0 2
   P5 -> 2 2 2 0 4
   P4 -> 2 2 2 2 4
   P3 -> 2 4 2 3 4
   P2 -> 4 4 3 3 4
   P1 -> 4 4 3 3 4