Sistemas Operativos
1° Rec 1° Parcial 1C2022 - Resolución

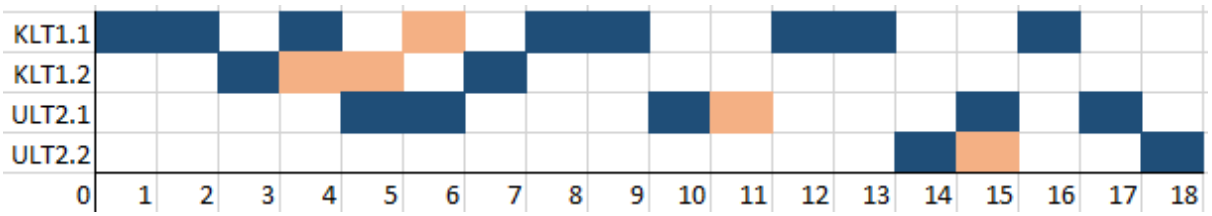
<u>Aclaración</u>: La mayoría de las preguntas o ejercicios no tienen una única solución. Por lo tanto, si una solución particular no es similar a la expuesta aquí, no significa necesariamente que la misma sea incorrecta. Ante cualquier duda, consultar con el/la docente del curso.

Teoría

1. Al usar directamente syscalls el código es más "performante", pero menos portable. A priori esperaríamos menos cambios de modo porque una buena parte del tiempo que se ahorra ahora viene de ahí.
2. Al comienzo, cuando no se protegían las secciones críticas existía el riesgo de que se produzca condición de carrera ocasionando resultados erróneos o poco confiables.
   Cuando se implementaron soluciones de software para resolver la mutua exclusión se logró que los resultados sean confiables, pero aumentó la tasa de uso de la CPU a causa de la espera activa.
   Al actualizar del uso de soluciones de software al uso de semáforos se logró evitar la espera activa mantenimiento los resultados confiables.
3.
   a. Falso. Los procesos son independientes entre sí y cada uno tiene su propio heap.
   b. Falso. Los procesos ULT deben compartir el quantum porque este es asignado al proceso, en cambio a los KLT se les asigna el quantum que el SO. De todas maneras, algunas justificaciones verdaderas se podrían tomar como válidas, pensando en que los ULTs tienen menos overhead en los cambios de contexto.
4. Esperamos tres colas: arriba una FIFO (para las llamadas), una intermedia para las apps que están corriendo activamente (podríamos pensar algún tipo de RR, por si hay varias abiertas al mismo tiempo) y otra de menor prioridad para minar y las otras apps (podría ser otro RR). Cada proceso entra y sale de su cola, pero al cerrar una app de la cola intermedia debería ir a la de baja prioridad.
5. El concepto de estado seguro hace referencia a que, dadas las matrices de peticiones máximas y de recursos asignados de un sistema, el algoritmo garantiza que no podrá producirse deadlock aún si en ese momento se peticionaran todos los recursos restantes posibles (matriz de necesidad). Este concepto no existe en el algoritmo de detección ya que el mismo no toma ninguna medida para que el deadlock no ocurra, sino que solamente lo detecta y luego intenta solucionarlo.

Práctica

1. a) Aclaración: En t=7 tendría que seguir ejecutando ULT2.1 en vez de KLT 1.1, ambos llegan a ready en t=6 (Fin de Q vs Fin de I/O)



b) El diagrama hubiese cambiado a partir del instante 6. Donde ejecutaría KLT2.2 por haber arribado en el instante 3 y se le hubiese asignado un Q=2 para ejecutar.

c) El diagrama hubiese cambiado a partir del instante 3. Cuando el proceso 1 se interrumpe por quantum vuelve a la cola de listos, pero al ubicarse antes del Proceso 2, vuelve a ser elegido para ejecutar. Además en el instante 2 el Proceso P1 fue desalojado, no hay replanificación entre hilos. Continúa ejecutando ULT1.1

2. a)

bombo = 1
hihatGolpeado = 0
golpesHihat = 0
mutexHihat= 1

Pie izquierdo	Pie Derecho	Mano Izquierda	Mano Derecha
W(hihatGolpeado, 3) Cerrar(hi hat) **S(golpesHihat, 8)** **W(hihatGolpeado, 8)** Abrir(hi hat) **S(bombo)**	**W(bombo)** PedalDerecho(bombo) **S(golpesHihat, 3)**	**W(golpesHihat)** **W(mutexHihat)** Golpear(hi hat) **S(mutexHihat)** **S(hihatGolpeado)**	**W(golpesHihat)** **W(mutexHihat)** Golpear(hi hat) **S(mutexHihat)** **S(hihatGolpeado)**

b) Conviene que estén implementados con espera activa si la ejecución de las instrucciones es rápida y por lo tanto, la espera de los semáforos cortas.

3. Matriz necesidad:
   P1: (2,1,1,0)
   P2: (1,0,0,3)
   P3: (1,2,0,0)
   P4: (0,1,0,0)

   Disponibles: (2,1,1,0)

a) Puede ejecutar y finalizar P4 liberando sus recursos asignados
   Disponibles queda: (2,2,1,0)

   Puede ejecutar y finalizar P3 liberando sus recursos asignados
   Disponibles queda: (2,2,1,2)

   Puede ejecutar y finalizar P1 liberando sus recursos asignados
   Disponibles queda: (3,3,2,3)

   Puede ejecutar y finalizar P2 liberando sus recursos asignados
   Disponibles queda: (4,3,4,3)

   Hay secuencia segura P4>P3>P1>P2, entonces hay estado seguro.

b) Se modifican las matrices para simular que al Proceso 2 se le asigna una instancia del recurso 1.

   Matriz necesidad:
   P1: (2,1,1,0)
   P2: (0,0,0,3)
   P3: (1,2,0,0)
   P4: (0,1,0,0)

   Disponibles: (1,1,1,0)

   Puede ejecutar y finalizar P4 liberando sus recursos asignados
   Disponibles queda: (1,2,1,0)

   Puede ejecutar y finalizar P3 liberando sus recursos asignados
   Disponibles queda: (1,2,1,2)

   No se obtiene una secuencia segura por lo tanto no es estado seguro y se rechaza la solicitud realizada por el proceso 2.