Sistemas Operativos
1° Parcial 2C2022 - TM - Resolución

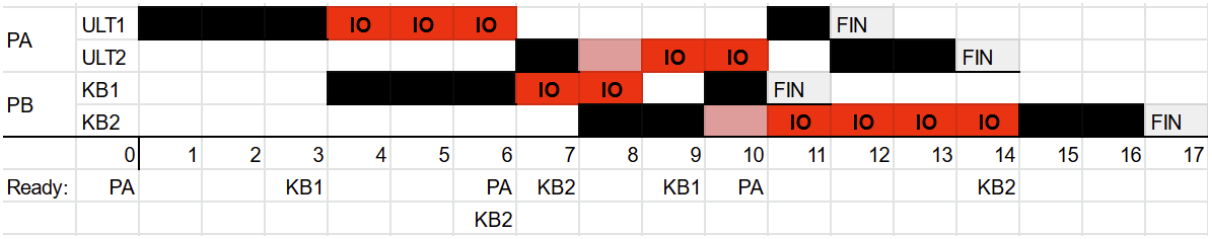
Aclaración: La mayoría de las preguntas o ejercicios no tienen una única solución. Por lo tanto, si una solución particular no es similar a la expuesta aquí, no significa necesariamente que la misma sea incorrecta. Ante cualquier duda, consultar con el/la docente del curso.

Teoría

1. Como mínimo un wrapper tendría dos, uno para ir a modo kernel, y otro para volver a modo usuario. La cantidad de cambios de modo depende de la cantidad de syscalls que realice dicha función (podrían ser más que 2 si se realizan varias syscalls en la misma función wrapper).
2. Falso. Al estar el sistema operativo ya atendiendo una interrupción, sólo sería necesario un cambio de contexto
3. Algoritmos: SJF (con/sin desalojo), HRRN, Feedback Multinivel. El sistema operativo solo puede estimar las ráfagas futuras mediante alguna fórmula como la de la media exponencial.
4. Estrategias
    a. Solución de software: genera espera activa
    b. Hardware: deshabilitar interrupciones: no genera espera activa
    c. Hardware: instrucciones especiales: requieren espera activa
    d. Semáforos / monitores: no generan espera activa
5. Tanto Prevención como Evasión serían estrategias válidas, dado que es obligatorio que el sistema no sufra ningún tipo de bloqueo permanente ni sacrificar su funcionamiento en un recupero. En caso de que no se pueda tolerar cierto overhead, Prevención sería mejor dado que Evasión implica un overhead en el análisis del estado del sistema que se realiza con frecuencia.

Práctica

1. a)



b) En el instante 3, al tener un Quantum de 4, PA debería seguir ejecutando ULT2 ya que no estaría bloqueado aun..
c) En el instante 7, PB no podría ejecutar aun ya que estaría bloqueado por la I/O iniciada por el ULT B1.

2.

Robot de almacén (N instancias)	Robot distribuidor (M instancias)	Camión (4 instancias)

while(1) {
 	
W(total_cajas)
caja = tomar_caja()
 	
W(espacio_en_plataforma)
W(m_plataforma)
depositar(caja, plataforma)
S(m_plataforma)
S(cajas_en_plataforma)
 	
}
	while(1) {
	
	W(cajas_en_plataforma)
	W(m_plataforma)
	caja = retirar(plataforma)
	S(m_plataforma)
	S(espacio_en_plataforma)
	
	despachar(caja, caja.destino)
	S(despachado[caja.destino])
	
	}
		W(despachado[get_id]) x 25
		salir()

m_plataforma = 1
total_cajas = 100
espacio_en_plataforma = 5
cajas_en_plataforma = 0
despachado = [0, 0, 0, 0]

3.

1) Primero calculemos la matriz de Necesidad:
   Corrección: Necesidad de P2 = [1,0,1,4]

	R1	R2	R3	R4
P1	1 -> 0	2 -> 1	0	0
P2	0	0	1	4
P3	1	2	4	0
P4	0	0	1	0

Mín 2, 2, 4, 4

- Totales (2,2,4,4) - asignado (1,0,3,2) = (1, 2, 1, 2)
- Ahora veamos si con ese vector, al asignar 1 1 0 0 el sistema queda en estado seguro:
    - Inicial: 1 2 1 2
    - Asigno petición P1 -> 0 1 1 2
    - Arranca el algoritmo:
        - Término de atender a P1 -> 2 2 1 3
        - Atiendo a P4 -> 2 2 2 3
        - No puedo atender ni a P2 ni a P3, como mínimo, necesitaría agregar una instancia de R4 -> 2 2 2 4
        - Atiendo P2 -> 2 2 4 4
        - Atiendo a P3 -> 2 2 4 5 => este sería el **mínimo** vector de recursos totales para que se cumpla la condición

2) Partiendo de recursos totales = 2 2 4 5 => rec disponibles = 1 2 1 3

- P3 pide 1 R1 -> 0 2 1 3
- Arranca algoritmo: P4 -> 0 2 2 3
- No puedo atender a ninguno más, dejaría el sistema en estado inseguro, no se asigna