Sistemas Operativos
1° Parcial 1 Rec 1C2024 - 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. a) F, si la interrupción se produjo durante la atención de otra interrupción, solamente es requerido un cambio de contexto y no de modo, ya que ambas se realizan en modo kernel, por ser el sistema operativo quien debe atenderlas.
   b) F, la ejecución de una syscall implica además un cambio de contexto, ya que se le solicita al sistema operativo que la ejecute por lo cual debe de hacerse un cambio de modo y de contexto para su correspondiente procesamiento, y al finalizar se ha de devolver el contexto al proceso y el modo debe de ser cambiado a modo usuario
2. En este caso, dado que sabemos que la función externa tiene un memory leak, y teniendo la posibilidad de correrla en un proceso o un hilo, la mejor opción es correrla en un proceso, ya que cuando la misma finalice, la memoria "leakeada" se devolverá ya que finalizará el proceso también.
   La desventaja de este enfoque es que la creación de un proceso es más costosa que la creación del hilo.
3. No se aprovecharía mucho el uso de jacketing ya que aunque el proceso no se bloquee tras una operación de I/O, lo más probable es que el mismo sea desalojado al poco tiempo por lo que no aprovecharía a ejecutar mientras se realiza la operación.
   Por otro lado, si el sistema operativo utiliza un algoritmo de planificación RR o si se produjera una interrupción durante la ejecución de un ULT que provoque desalojar al proceso de la CPU, el estado del mismo pasaría de Running a Ready, pero el ULT dado que no es conocido por el sistema operativo, no tiene manera de actualizar su estado, quedando en Running a pesar de que su proceso se encuentre en Ready.
4. Dos hilos accediendo a una variable compartida puede traer condiciones de carrera si el acceso no es sincronizado correctamente.
   Para que sea un problema, al menos uno de los hilos tiene que estar modificando la variable y tiene que haber posibilidad de concurrencia en el acceso a la misma.
5.

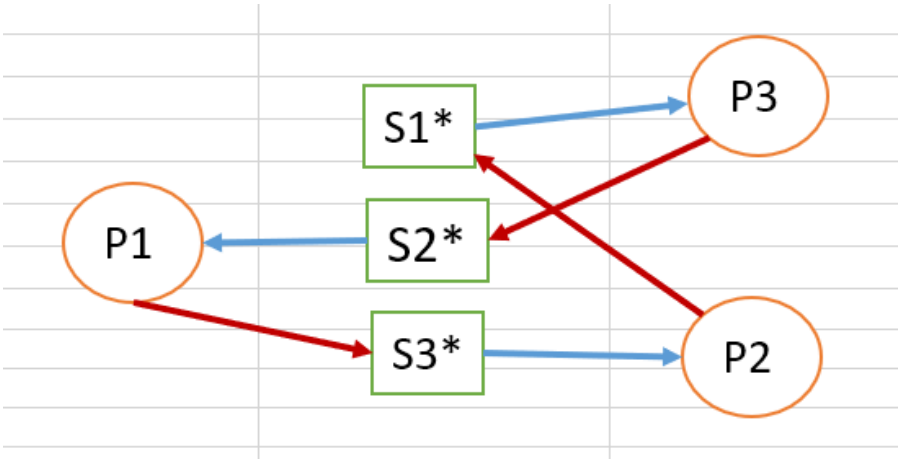
	Prevención	**Evasión**	**Detección**
Overhead	Bajo (son políticas definidas a priori)	Alto (corre el algoritmo en cada petición)	Medio (depende de la frecuencia con la que se corra el algoritmo)
Puede ocurrir deadlock	No	No	Sí
Flexibilidad a la hora de realizar peticiones	Baja, limitada por la política definida	Medio (cada proceso debe declarar recursos máximos)	Alta (cualquier proceso pide recursos libremente)

Práctica

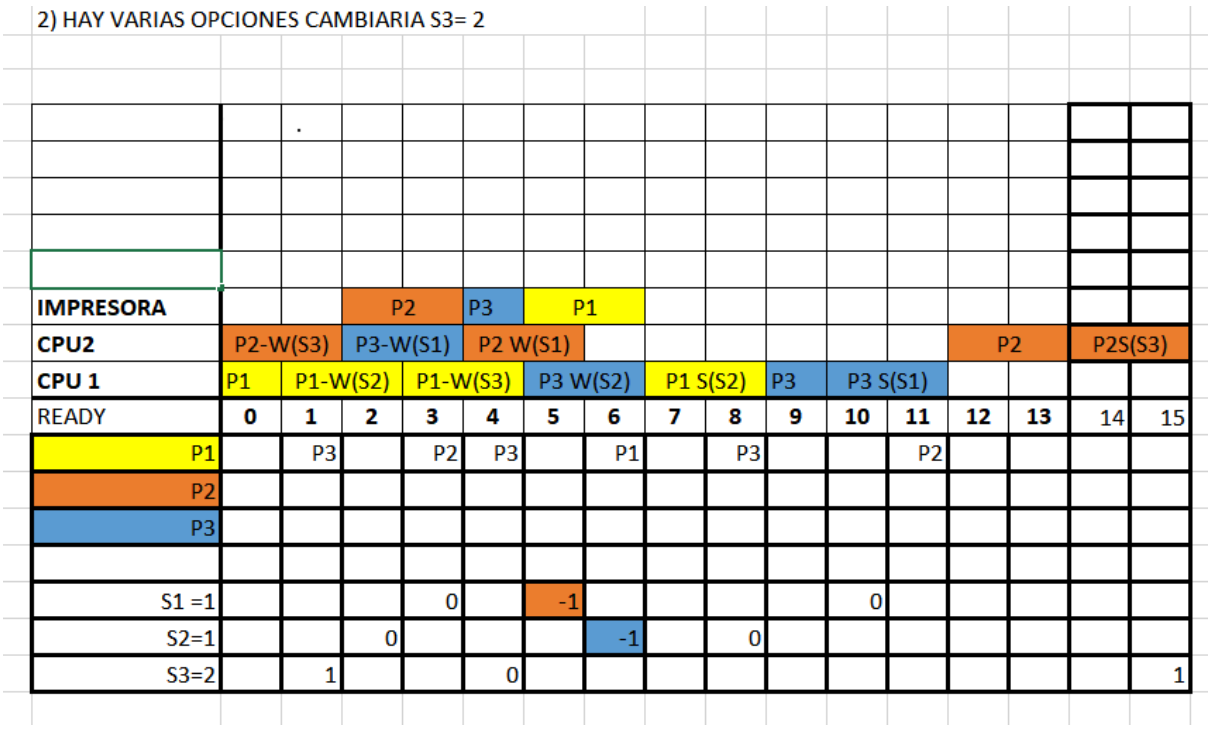
1.

IMPRESORA			P2		P3			
CPU2	P2-W(S3)		P3-W(S1)		P2 W(S1)			
CPU 1	P1		P1-W(S2)		P1-W(S3)		P3 W(S2)	
READY	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
P1		P3		P2	P3			
P2								
P3								
S1 =1				0		-1		
S2=1			0				-1	
S3=1		0			-1			

1) P1-P2 Y P3 ESTAN DEADLOCK



2) HAY VARIAS OPCIONES CAMBIARIA S3= 2



2.

turno[5] = {0, 0, 0, 0, 0}

dibuPosicionado, **pateadorPosicionado**, **patear**, **pateó**, **atajó**, **validado**, **reaccionado** = 0

Árbitro (1 instancia)	Pateadores (5 instancias)	Dibu (1 instancia)

DarTurno(pateador_id)
SIGNAL(turno[pateador_id])
WAIT(dibuPosicionado)
PitarSilbato()
SIGNAL(patear)
WAIT(atajó)
GOL = validarGol()
SIGNAL(validado) x 2
pateador_id = next()
WAIT(reaccionado) x 2
	WAIT(turno[getID()])
	IntentarNoMirarAlDibu()
	Posicionarse()
	SIGNAL(pateadorPosicionado)
	WAIT(patear)
	Patear()
	SIGNAL(pateó)
	WAIT(validado)
	If (GOL == FALSE)
	  inventarExcusa()
	SIGNAL(reaccionado)
		WAIT(pateadorPosicionado)
		EstudiarJugador(pateador_id)
		Posicionarse()
		SIGNAL(dibuPosicionado)
		WAIT(pateó)
		Atajar()
		SIGNAL(atajó)
		WAIT(validado)
		If (GOL == FALSE)
		  baile()
		SIGNAL(reaccionado)

3.

a) RR (Q=2). Puede verse dado que los KLTs son desalojados sin algún otro evento aparente cada 2 instantes de tiempo como en t=6, t=8, t=10.
b) SJF con desalojo (SRT). KLT1 comienza eligiendo a ULT2 por ser más corto en t=4, el desalojo puede verse en t=17 donde llega ULT4 y desaloja a ULT3 por ser más corto que la remanente ráfaga.
c) Utiliza jacketing. Si no utilizara, el GANTT cambiaría a partir del instante t = 9.