Sistemas Operativos
1° Rec 1° Parcial 1C2023 - 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.
    a. Verdadero. Esto se debe a que corren en modo usuario, y si intentaran ejecutar una instrucción privilegiada se produciría una interrupción indicando un error.
    b. Falso. Un proceso que puede modificar el PC de forma malintencionada podrá disponer de todo el código contenido en el espacio de direcciones del mismo, pero el hardware validará cualquier intento de acceso por fuera del mismo y lanzará una interrupción que derivará en un error.

2. Los algoritmos con desalojo tienen en cuenta los mismos eventos de los algoritmos sin desalojo (bloqueo del proceso por syscall y finalización del mismo), y consideran también el evento que derive en un ingreso a la cola de Ready (syscalls de creación de hilos/procesos, o interrupciones que indiquen fin de ráfaga o fin de I/O).
   Los algoritmos sin desalojo descartan estos últimos eventos porque deciden no expropiar la cpu al proceso una vez que la misma fue asignada.

3. Los algoritmos de planificación que no sufren starvation y minimizando su overhead son FIFO y Round robin, ya que FIFO planifica los procesos según van situándose en orden a la cola de ready siendo ejecutados en el mismo orden en que fueron ingresados y es el algoritmo con menos overhead debido a que es el más básico, por otro lado RR funciona como FIFO con el agregado de mantener cierta equidad en el uso de la CPU enviando interrupciones cada vez que un proceso agote su Quantum.
   En FIFO el sistema operativo se involucra en los instantes donde se ejecute alguna syscall que provoque un bloqueo del proceso y durante la finalización del mismo, mientras que en RR además interviene cada vez que se lance una interrupción por fin de Quantum debido a que esta debe ser atendida para desalojar al proceso en cuestión.

4. Utilizar semáforos mutex provistos por el SO implicaría que ante un wait() se bloquee todo el Proceso/KLT y por lo tanto el mismo nunca se desbloquee si el signal() se suponía que lo haga uno de sus ULTs pares.

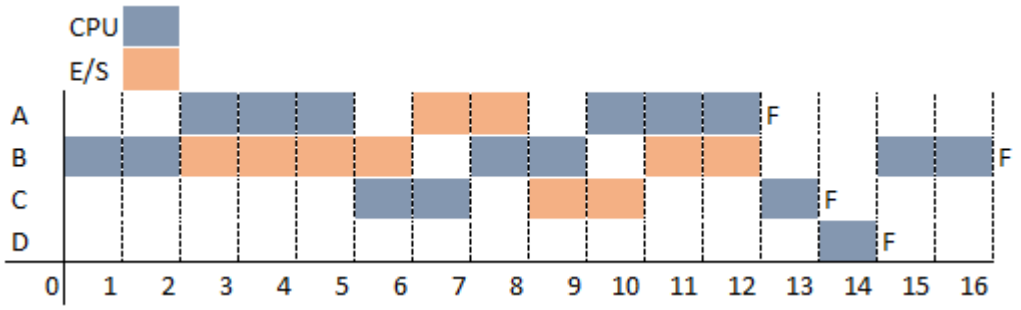
5.

Overhead generado:
La estrategia de prevención es la que menos overhead genera ya que son políticas del SO definidas a priori. La estrategia de evasión requiere que ante cada petición de recursos se ejecute el algoritmo del banquero, con lo cual puede ser bastante alto. En la estrategia de detección y recuperación el overhead es causado por la ejecución del algoritmo para detectar deadlock, con lo cuál depende de cada cuánto se corra el mismo. Además hay un overhead asociado a cada posible estrategia de recuperación, siendo menor en los casos en que se finalizan todos los procesos y siendo mayor en los casos en que se elige una víctima en particular ya que hay un costo asociado a la hora de seleccionarla y otro a la hora de verificar que realmente se haya solucionado.

Limitaciones a la hora de pedir recursos:
La estrategia menos restrictiva es la de detección y recuperación ya que no limita de ninguna manera la forma o la cantidad de recursos a pedir. Evasión requiere que se declaren de antemano la cantidad máxima de recursos a utilizar con lo cuál de alguna forma restringe la cantidad de recursos que un proceso puede pedir. En este aspecto la estrategia más restrictiva es la de prevención, que dependiendo de la política podría forzar a que los recursos deban ser pedidos en un orden específico, por ejemplo.

Práctica

1. a)



Cálculo de la tasa de respuesta:
T=2
A: (1+3)/3 = **1,33**
C: (0+2)/2 = 1

T=12
B: (0+2)/2 = 1
C: (2+1)/1 = **3**
D: (0+1)/1 = 1
NOTA: D llega al sistema en t=9, pero ingresa a Ready en t=12

T=13
B: (1+2)/2 = 1,5
D: (1+1)/1 = **2**

b) Hasta la llegada del Proceso D no hay cambios. Luego es necesario analizar cada decisión del planificador.
Cálculo de la tasa de respuesta:
T=9
A: (1+3)/3 = **1,33**
D: (0+1)/1 = 1
No hay cambios

T=12
B: (0+2)/2 = 1
C: (2+1)/1 = 3
D: (3+1)/1 = **4**
En el instante 12, el diagrama se modifica porque debe ejecutar el proceso D

2. usuarios_conectados=0, conexiones_entrantes=20, mtx_conectados=1, mtx_en_evaluacion=1, solicitar_archivo=0, archivo_creado[N]={0,0,0,....}

Conexiones_Nuevas(1)

while(TRUE){
usuario=recibir_conexion()
wait(conexiones_entrantes)
wait(mtx_conectados)
agregar_usuario(usuario, conectados)
signal(mtx_conectados)
signal(usuarios_conectados)
}

Generador_Informe (1)	**Evaluador (N)**

while(TRUE){
wait(solicitar_archivo)
wait(mtx_en_evaluacion)
usuario=obtener_usuario(en_evaluacion)
signal(mtx_en_evaluacion)
crear_archivo(usuario)
signal(archivo_creado[obtener_evaluador(usuario)])
}
	while(TRUE){
	wait(usuarios_conectados)
	wait(mtx_conectados)
	usuario=obtener_usuario(conectados)
	signal(mtx_conectados)
	asignar_evaluador(usuario, id_evaluador)
	wait(mtx_en_evaluacion)
	agregar_usuario(usuario, en_evaluacion)
	signal(mtx_en_evaluacion)
	signal(solicitar_archivo)
	wait(archivo_creado[id_evaluador])
	informe=obtener_archivo(usuario)
	vulnerabilidades = evaluacion_vulnerabilidades(usuario)
	escribir_archivo(vulnerabilidades, informe)
	enviar_informe(usuario, informe)
	signal(conexiones_entrantes)
	}

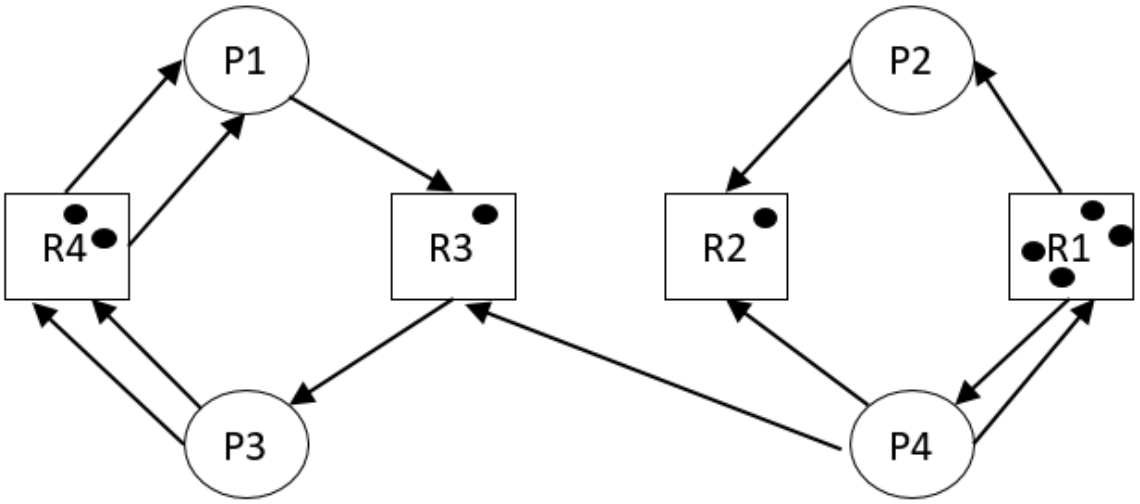
3.

Recursos Disponibles (2,1,0,0)
Inicia el Algoritmo de Detección:
Ejecuto Proceso 2 y libera sus recursos --> Disponibles (3,1,0,0)
Deadlock entre Procesos 1, 3 y 4.
Si se elimina el Proceso 4, no se soluciona el Deadlock. Debe eliminarse el Proceso 1 ó 3.

*Opción 1:
Elimino Proceso 1 y se liberan sus recursos --> Disponibles (3,1,0,2)
Ejecuto Proceso 3 y libera sus recursos --> Disponibles (3,1,1,2)
Ejecuto Proceso 4 y libera sus recursos --> Disponibles (4,1,1,2)

*Opción 2:
Elimino Proceso 3 y se liberan sus recursos --> Disponibles (3,1,1,0)
Ejecuto Proceso 1 y libera sus recursos --> Disponibles (3,1,1,2)
Ejecuto Proceso 4 y libera sus recursos --> Disponibles (4,1,1,2)

b)



Según el punto a) los procesos 1, 3 y 4 están en deadlock. Pero en el grafo puede visualizarse que el proceso 4 no participa de ningún ciclo con otro proceso, por lo tanto, no forma parte del Deadlock. Los procesos en deadlock son los procesos 1 y 3.

c) Agregando una instancia de R3, no habría deadlock. Si se agregara una instancia de los Recursos 1, 2 o 4, el deadlock se produciría igual.