Sistemas Operativos
1° Parcial 1C2017 - TT - 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 profesor del curso.

Teoría

1) Herramienta utilizada por los procesos para solicitar servicios al sistema operativo que de otra forma le serían imposibles poder realizar, como el uso de recursos de hardware (leer disco por ejemplo) o de otra índole (crear un klt por ejemplo).
Los procesos ejecutan en modo usuario, bajo el cual no se pueden ejecutar instrucciones privilegiadas. Al realizar una llamada al sistema, se cambia a modo kernel, y solamente en dicho modo se pueden ejecutar instrucciones privilegiadas.

2) Convendría usar ULT por sobre KLT cuando se requiera overhead al mínimo (debido a que no hay cambios de modo), o se requiera portabilidad del programa (dado que sobre bibliotecas estándares podria usarse en cualquier SO), o se requiera una planificación particular no provista por el sistema operativo.
Dos atributos pueden ser el TID (thread id) y el contexto de ejecución (PC, flags, etc)

3) El sistema operativo solo puede recuperar el control de la CPU cuando el proceso finaliza o se bloquea, por lo que un usuario/proceso podría nunca ejecutar si otro proceso nunca libera el procesador, ya sea por un error o porque deliberadamente no se bloquea nunca.

4) Los mismos garantizan la mutua exclusión a través de alguna solución de software (algoritmo de peterson por ejemplo) o usando la instrucción de hardware (como test_and_set). Tienen la ventaja de sirven para sistemas multiprocesador sin generar overhead extra (como sí lo haría el deshabilitar las interrupciones). Tienen la desventaja de generar una pequeña carga de espera activa.

5) Estado seguro significa que se tiene la certeza de que nunca ocurrirá un deadlock y que existe al menos una secuencia segura de finalización de los procesos.
El sistema nunca podría quedar en estado inseguro, debido a que la estrategia consiste en no otorgar recursos que pudieran dejarlo en dicha situación.

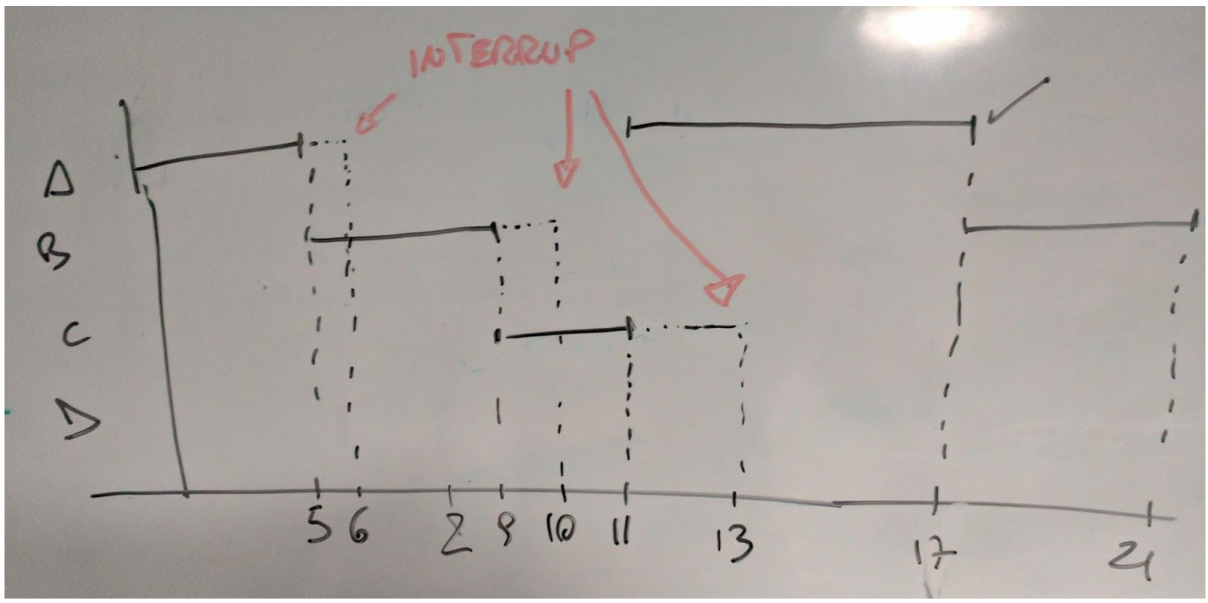
Práctica

Ejercicio 1

Con alpha = 0,4 los valores dan:

llegada	inicial-e	inicial-r	cpu e	cpu r	i/o	cpu e	cpu r
A - en ready	2	2	2	5	1	3,2	6
B - en ready	3	3	3	4	1	3,4	4
C- 8	1	1	1	2	2	-	-
D- nunca ejecuta	-	-	-	-	-	-	-

Parte a):



Nótese como C nunca ejecuta su última ráfada por haber sido finalizado, y D ni siquiera empieza a ejecutar, por pertenecer al mismo Proceso que C.

Parte b)
No fueron adecuadas las estimaciones, dado que tuvieron un gran margen de error en los casos de los KLTs A y B. El KLT A empezó a ejecutar ráfagas más grandes, y sin embargo el sistema operativo al estimar siguió calculando que dichas ráfagas serían más chicas que las de KLT B (cuando no fué así).
El problema se debe a que el valor alpha de 0,4 tiende a priorizar más la historia general de ejecución, y no tanto las rágafas reales más recientes. Por lo tanto las estimaciones tardan en adaptarse a los nuevos tamaños de ráfagas.
Al probar con un alpha mayor o igual a 0,6 la adaptación se produce más rápido, permitiendo que B (con cpu estimada en 3,6) pueda al menos ejecutar su última ráfaga antes que A (con cpu estimada en 3,8).

Ejercicio 2

freno = anteojos = 1; <el resto> = 0

Pie Izquierdo	Pie Derecho	Mano derecha	Mano Izquierda
wait(freno) if (paredEnfrente) frenar() signal(freno) pisar_embrague() signal(primera) wait(soltar_em) soltar_embrague() signal(acelerar)	wait(freno) if (paredEnfrente) frenar() signal(freno) wait(acelerar) pisar_acelerador()	wait(primera) poner_cambio() signal(volante_iz) wait(anteojos) acomod_anteojos() signal(anteojos) agarrar_volante()	wait(anteojos) acomod_anteojos() signal(anteojos) wait(volante_iz) agarrar_volante() signal(soltar_em)

Ejercicio 3

1) Primero calculemos la matriz de Necesidad

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

Los recursos totales asignados son: 1, 0, 3, 2 => rec disponibles = 1, 2, 1, 3

Asigno: 0 1 1 3

Termino de atender a P1 -> 2 2 1 4
Atiendo a P4 -> 2 2 2 4
Atiendo P2 -> 2 2 4 4
Atiendo a P3 -> 2 2 4 5 -> pude encontrar una secuencia segura, por lo tanto la asignación dejaría al sistema en un estado seguro. Conclusión: se otorgará el recurso sin problemas.

2).
Teniendo (1,2,1,3) como disponibles, y pensando en la matriz de máximos como la matriz de solicitudes pendientes:

- Ninguno de los procesos actuales pueden finalizar
- Por lo tanto, todos están en deadlock.