Sistemas Operativos 1° Parcial 2C2017 - 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. Un proceso nunca es seleccionado para ejecutar en el procesador debido a la constante llegada a Ready de procesos de mayor prioridad. Algoritmos que podrían sufrir inanición:
 - Por prioridades fijas: se puede solucionar utilizando aging, aumentando con el tiempo la prioridad hasta que un proceso se pueda ejecutar (transformándolo en prioridades dinámicas).
 - SJF: no puede solucionarse a menos que se modifique el algoritmo (por ejemplo, teniendo en cuenta la variante tiempo de espera, convirtiéndose en HRRN)
 - Feedback multinivel: se puede subir de cola a los procesos, considerando el tiempo de espera

2. V o F

- 1. Falso. La syscall se encarga de realizar el cambio de modo. Los wrappers permiten abstraer de detalles de bajo nivel de las syscalls y ganar portabilidad.
- 2. Falso. Se podría estar ejecutando una syscall y que llegue una interrupción que haya que atender, se cambiaría el contexto del procesador pero no necesariamente se cambiaría el proceso (por ejemplo en planificación sin desalojo).
- 3. No sería correcto, porque los bugs ocasionados por problemas de condiciones de carrera dependen de la velocidad relativa de ejecución de dos programas, y esta es explícitamente modificada mientras se debuguea (debido a que se frena la ejecución de alguno de ellos de forma temporal) pudiendo generar condiciones bajo las cuales nunca ocurriría el problema.
- 4. Los ults ofrecen portabilidad, planificación a medida y menor overhead que los KLT. En contrapartida, los mismos no permiten paralelismo (ULTs del mismo proceso) y al bloquearse uno, se bloquean todos los del mismo KLT (excepto que se esté utilizando Jacketing).
- 5. Se puede administrar mediante:
 - Una lista de bloques libres
 - Tener los bloques libres enlazados
 - Un bitmap de bloques libres

Práctica

Ejercicio 1

1)

FAT					
0			ptr dir	5	bloque X
1			ptr dir	4	8
2			ind sim	bloque x	- -
3			ind dob	_	
4	8		ind tri		
5	4				
6					
}	FFFF(FIN)				
		1			

2) FS A: 1MiB -> 1MiB/ 4KiB = 256 clusters -> 256 -3 = 253 clusters extra.

FS B: 1MiB -> 1MiB/ 1KiB = 1024 bloques de datos.

PtrsXBloque = 1KiB $/8b(64bits) = 2^10/2^3 = 2^7 = 128 ptrs por bloque$

=> con el ptr ind simple se apunta a un bloque de punteros que direcciona 128 bloques de datos... se requieren direccionar a 1024 - 2 - 128 = 894 bloques más.

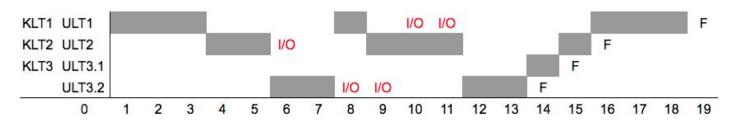
Se va a requerir un bloque de punteros que es el apuntado por el ptr de doble indirección .. además se requerirán bloques de punteros de segundo nivel para apuntar a esos 894 bloques restantes.

- => 894/128 = 6,9... -> 7 bloques de punteros extra => se requieren 1024 bloques de datos y (1 ind.simple + 1 ind.dobleNivel1 + 7 ind.dobleNivel2) bloques de punteros = 1033 bloques. Como ya tenía asignados 4 bloques (3 datos y 1 de ptrs) se requerirán 1029 bloques extras.
- 3) FS A -> máx $2^16 * 4KiB = 2^16 * 2^12 = 2^28 = 256MiB$. Por lo tanto el MIN(256MiB, 512MiB) = 256 Mib máximo real.

FSB -> $[2 + 1 \times 1 \text{kb/8b} + 1 \times (1 \text{kb/8b})^2 + 1 \times (1 \text{kb/8b})^3] \times 1 \text{kb} = 2 \text{ gb apróx}$. MIN(2GiB, 512MiB) = 512 MiB máximo real.

Ejercicio 2

a)



El ULT3.2 finaliza bien, y luego se finaliza el KLT3.

b) Instantes = {0, 1, 2, 3, 5, 6 7, 8, 9 10, 11, 13, 14, 17}.

Ejercicio 3

saxos_seguidos = 3, bajo_responde = 0, improvisar = 0, turno_guitarra = 1, turno_piano = 0

Saxo (5 instancias)	Bajo (1 instancia)	Guitarra (1 instancia)	Piano (1 instancia)
while (true){ wait(saxos_seguidos) tocar_una_melodia() signal(bajo_responde) }	while (true){ wait(bajo_responde) responder_melodia() signal(saxos_seguidos) signal(improvisar) }	while (true){ wait(turno_guitarra) wait(improvisar) improvisar() signal(turno_piano) }	while (true){ wait(turno_piano) wait(improvisar) improvisar() signal(turno_guitarra) }