Final de Sistemas Operativos

16/07/2019

Nota:

Apellido y Nombre	*Profesor*	*Tomé conocimiento de la nota: (Sólo aplazos)*

Preguntas teóricas					Ejercicios	
1	2	3	4	5	1	2

A) Teoría: Defina explícitamente como VERDADERA o FALSA cada una de estas afirmaciones justificando brevemente.

1) En un sistema que sufre el fenómeno de inversión de prioridades, es posible solucionar este problema cambiando su algoritmo de planificación por Virtual Round Robin.

2) En un entorno con un procesador poco potente, muchos recursos y la necesidad de garantizar que no haya Deadlocks, conviene implementar una estrategia de Evasión.

3) Un File System que utiliza asignación contigua atiende lecturas secuenciales con rapidez. Un esquema FAT podría alcanzar igual rapidez, en ciertos casos.

4) Los KLTs no pueden ocasionar memory leaks, dado que el TCB no tiene referencia al heap del proceso.

5) Toda E/S síncrona debe ser bloqueante.

B) Práctica: Resuelva los ejercicios justificando las respuestas

1) Un Sistema Operativo usa segmentación paginada, con memoria virtual, punteros de 16 bits y marcos de 4 KB y una TLB de 4 entradas. La tabla de segmentos de un proceso tiene la siguiente información, siendo RWX permisos de lectura, escritura y ejecución. Los procesos no pueden tener más de 4 segmentos, y estos son los únicos de este determinado proceso.

Segmento 0 | RW-		Segmento 1 | -X-		Segmento 2 | R--		TLB	
Frame	P | U	Frame	P | U	Frame	P | U	Dir	Frame
6	1 | 0	9	1 | 0	A	0 | 0	B	8
9	0 | 0	B	1 | 1	1	0 | 0	3	1
11	1 | 1	7	1 | 0	C	0 | 0	(el resto está vacío)	
(tiene más páginas)		(tiene más páginas)		(tiene más páginas)			

a) Indique las direcciones que generarán las siguientes escrituras del proceso en cuestión: 3C38h, FFACh, AA00h, sabiendo que el algoritmo de reemplazo de páginas es Clock, con reemplazo global y asignación dinámica.

b) Se cree que uno de los segmentos contiene una biblioteca compartida, usada con muy poca frecuencia. Indique si esto podría o no ser así y porqué.

2) Tres tipos de procesos se utilizan para generar y leer reportes de una base de datos de gran tamaño. Los diferentes generadores de reportes obtienen una de las 10 conexiones disponibles, y luego de crear la consulta que necesitan correr, la envían al motor de base de datos. Una vez obtenida esa información generan reportes, y una vez listos los escriben en un gran archivo de reportes (siendo esta la única estructura compartida entre procesos). Una serie de proceso lectores consume luego dichos reportes, sin borrarlos del archivo en cuestión. Dado que va creciendo cada vez más, las lecturas sobre el archivo de reportes pueden llevar horas, pero las escrituras están optimizadas para llevar unos pocos segundos.

Lector (10 instancias)	Generador de reportes (100 instancias)	Motor de DB (1 instancia)

`while (1) {`
`  nro = rand()`
`  rep = leer(reportes, nro)`
`  print(rep)`
`}`
	`while (1) {`
	`  db = obtenerConexion()`
	`  query = generarConsulta()`
	`  data = correr(query, db)`
	`  rep = generar(data)`
	`  escribir(reportes, rep)`
	`}`
		`while (1) {`
		`  abrirConexion()`
		`  devolverConsulta()`
		`  cerrarConexion()`
		`}`

a) Sincronice los procesos para que funcionen correctamente, utilizando sólo semáforos

b) Indique si en alguno de los casos, sería útil utilizar otra herramienta en lugar de semáforos, explicando porqué

El tiempo de duración del examen final será de 90' a contar desde el momento de comienzo del mismo. Si el alumno por algún motivo comenzará más tarde sólo podrá utilizar el tiempo remanente. Utilice hojas separadas para la teoría / ejercicios.

Final de Sistemas Operativos

16/07/2019

Nota:

Resolución

Teoría

1) Verdadero. La inversión de prioridades se da cuando un proceso con menor prioridad retiene un recurso que necesita uno de mayor prioridad. El de mayor prioridad no puede continuar y el otro no ejecuta por su prioridad baja. En un VRR, eventualmente van a ejecutar todos un quantum, solucionando este problema eventualmente.

2) Falso. Teniendo muchos recursos y un procesador poco potente, la evasión generaría mucho overhead, con lo cual es recomendable utilizar prevención.

3) Falso. Si bien se podría pensar que los si bloques estuvieran contiguos (por ejemplo, post defragmentación), la velocidad de lectura sería similar, se debe considerar los N accesos a FAT que se van a haciendo antes de leer cada bloque.

4) Falso. Si bien no tienen una referencia directa, comparten el heap. Por ende, si pidieran memoria y nunca la devolvieran, generarían memory leaks.

5) Falso. También podría ser no bloqueante, si su implementación devolviera un error ante una espera prolongada. Ejemplo: Una e/s para leer de un archivo que no existe podría devolver un error, pero la misma lectura sobre un archivo existente podría "detener" la ejecución del proceso hasta que se finalice..

Práctica

1) a)
Los marcos tiene 4KB de tamaño → 12 bits para el offset.
Los segmentos son máximo 4 → 2 bits para el segmento -> 2 bits para páginas -> 4 páginas por segmento máximo

Tomamos el primer dígito hexa, lo pasamos a binario y separamos segmento y página

3 | C38h → 3 => 00| 11 ⇒ Segmento 0, Página 3 ⇒ No podemos verlo en la tabla de páginas pero está en la TLB ⇒ Frame 1
F | FACh → F => 11 | 11 ⇒ Segmento 3, Página 3 ⇒ El proceso no tiene segmento 3 ⇒ Acceso inválido
A | A00h → A => 10 | 10 ⇒ Segmento 2, Página 2 ⇒ No presente, pero intenta escribir un segmento de lectura ⇒ Acceso inválido

b) Podría ser el segmento 2, ya que es sólo lectura y sus páginas no están presentes en memoria principal, algo esperable para una biblioteca que se usa muy poco.

2) **`mut_rep = 1; conexiones = 10; pedido_con = 0; rta_con = 0; pedido_consulta = 0; rta_consulta = 0;`**

Lector (10 instancias)	Generador de reportes (100 instancias)	Motor de DB (1 instancia)

`while (1) {`
`  nro = rand()`
`  **wait(mut_rep)**`
`  rep = leer(reportes, nro)`
`  **signal(mut_rep)**`
`  print(rep)`
`}`
	`while (1) {`
	`  db = obtenerConexion()`
	`  **signal(pedido_con)**`
	`  **wait(rta_con)**`
	`  query = generarConsulta()`
	`  **signal(pedido_consulta)**`
	`  data = correr(query, db)`
	`  **wait(rta_consulta)**`
	`  rep = generar(data)`
	`  **wait(mut_rep)**`
	`  escribir(reportes, rep)`
	`  **signal(mut_rep)**`
	`}`
		`while (1) {`
		`  **wait(pedido_con)**`
		`  **wait(conexiones)**`
		`  abrirConexion()`
		`  **signal(rta_con)**`
		`  **wait(pedido_consulta)**`
		`  devolverConsulta()`
		`  **signal(rta_consulta)**`
		`  cerrarConexion()`
		`  **signal(conexiones)**`
		`}`

Algunos semáforos podrían estar una línea antes o después, dependiendo de la interpretación de cada estudiante.

b) El mutex usado para el archivo de reportes podría ser un lock del file system. De esta forma, las lecturas podrían hacerse concurrentemente.

El tiempo de duración del examen final será de 90' a contar desde el momento de comienzo del mismo. Si el alumno por algún motivo comenzará más tarde sólo podrá utilizar el tiempo remanente. Utilice hojas separadas para la teoría / ejercicios.