Final de Sistemas Operativos

23/02/2016

Nota:

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

Preguntas teóricas					**Ejercicios**	
1	**2**	**3**	**4**	**5**	**1**	**2**

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

1) El mapeo de archivos a memoria permite a los procesos manejar archivos muy grandes de manera eficiente y sencilla

2) Usando semáforos implementados con espera activa no hay posibilidad de deadlocks, ya que los procesos no se bloquean esperando para ingresar a una región crítica.

3) Si una system call se ejecuta de forma no bloqueante, igualmente realizará al menos un cambio de modo.

4) Lecturas secuenciales de archivos muy grandes requieren mayor cantidad de accesos a bloques en EXT que en FAT (asumiendo archivos y bloques de igual tamaño).

5) El algoritmo SJF con estimadores no puede aplicarse si el comportamiento de los procesos es muy errático

B) Práctica: Resuelva los ejercicios justificando las respuestas

1) En un sistema se ponen a correr 3 instancias del proceso Contador (c1, c2 y c3), una de Acumulador y una de Notificador. Contador obtiene el nombre de archivo de una cola de archivos pendientes (inicialmente con 20 nombres), lo lee de disco y cuenta cada ocurrencia de cada palabra, colocando dicho resultado en una cola compartida, "resultados". Acumulador, por otro lado, obtiene dichos resultados y los procesa, escribiéndolos en un archivo "resultadosFinales". Cuando se procesa un lote de 10 resultados, se determina cuál es la palabra con más ocurrencias, actualizando "maxOcurrencias" (que es compartida).

a. Utilizando VRR con un q = 3 planifique la ejecución de los procesos si los mismos llegan en los instantes tc1 = 0, tc2 = 3 , tc3 = 4, ta = 1, tn = 2. Indique en qué momento finaliza cada uno y si ocurrió algún problema en la ejecución.
b. Sincronice utilizando semáforos (bloqueantes) e indique qué cambiaría en la planificación.

C (contador)

```
var nombre = próximoArch(archivosPendientes); (1 UT)
var archivo = leer(nombre); (3 UT)
var resultadoParcial = contar(archivo); (4 UT)
colocar(resultadoParcial, resultados); (1 UT)
```

N (Notificador)

```
var mensaje = armarMensaje(maxOcurrencias); (1 UT)
notificar(mensaje); (2 UT)
```

A (acumulador)

```
while (1) { (1 UT)
  var resultado = obtener(resultados);  (1 UT)

  contResultados ++; (1 UT)
  var  resultadoFinal = procesar(resultado); (3 UT)

  escribir(resultadoFinal, resultadosFinales); (1 UT)

  if (contResultados >10) { (1 UT)
     maxOcurrencias = maxOcurrencias(resultadosFinales); (5 UT)
     finalizar(); (1 UT)
  }
}
```

Notas:
a. En el caso de intentar obtener un elemento de una lista vacía, ocurrirá una excepción y el proceso será finalizado.
b. En VRR los procesos son promovidos a la cola de mayor prioridad únicamente si les quedó un remanente de su quantum.

2) Un Sistema Operativo utiliza Segmentación simple, con un máximo de 16 segmentos por proceso y direcciones de 16 bits. Está ejecutando un proceso cuya tabla de segmentos es la de la derecha.

	Base	Limite
0	1000h	FFFh
1	7777h	1Ah
2	FA00h	FEh

a) Indique la dirección física que generarán los pedidos: 20FCh y 1A33h.

b) Indique la dirección lógica correspondiente a las siguientes direcciones físicas: 1ACCh y 70A1h.

c) Indique qué problema tendría utilizar memoria virtual en este esquema. Indique cómo podría solucionarse, manteniendo la segmentación.

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 comenzara más tarde sólo podrá utilizar el tiempo remanente. Utilice hojas separadas para la teoría / ejercicios.

Final de Sistemas Operativos

23/02/2016

Nota:

TEORÍA

1) Verdadero. Se debe a que esta funcionalidad expone a los procesos el sistema de memoria virtual del sistema operativo, aplicado a archivos comunes, el cual se encuentra altamente probado para ser muy eficiente. Es sencillo porque permite manejar el archivo como si fuera un conjunto de bytes en memoria, que se cargan y se persisten en el disco por sí solos.

2) Falso. Los procesos pueden bloquearse esperando otros recursos y cumplir las 4 condiciones.

3) Verdadero. El cambio de modo se realiza para que el SO atienda la syscall.

4) Verdadero. En archivos grandes es necesario leer los bloques de punteros, aunque la diferencia de lecturas es baja en proporción.

5) Falso. Si son muy erráticos (varían su comportamiento entre i/o bound y cpu bound), es útil que poner un valor de alpha tal que privilegie la historia reciente de los procesos por sobre la media de toda la historia.

PRÁCTICA

1a.)

C1		IO	IO	IO									X										
C2									IO	IO	IO									X			
C3												IO	IO	IO									X
A				error																			
N						X																	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	21
c1	a	n	c2	c3																			
RR	A	N	N C2	C2 C3			C2 C3	C3 C1	C1					C2		C2 C3							
FIFO				C1							C2			C3									

Los problemas que ocurren son:

- Como no está sincronizado el productor consumidor, el proceso A es finalizado ya que intenta obtener elementos antes de que los mismos sean ingresados en la cola
- El proceso N notifica un mensaje erróneo ya que no espera a que maxOcurrencias sea actualizada

Por los problemas de condición de carrera previamente vistos, vamos a sincronizar el código

1b.)

C (contador)

```
w(semArchivosPendientes)
var nombre = próximoArch(archivosPendientes); (1UT)
var archivo = leer(nombre); (3UT)
var resultadoParcial = contar(archivo); 4 (UT)

w(mResultados)
colocar(resultadoParcial, resultados); (1UT)
s(mResultados)
s(hayResultados)
```

N (Notificador)

```
w(semMaxOcurrencias)
var mensaje = armarMensaje(maxOcurrencias); ( 1 UT )
notificar(mensaje); ( 2 UT)
```

A (acumulador)

```
while (1) { (1 UT)

  w(hayResultados)
  w(mResultados)
  var resultado = obtener(resultados);  (1 UT)
  s(mResultados)

  contResultados ++; (1 UT)
  var  resultadoFinal =      procesar(resultado); (3 UT)

  escribir(resultadoFinal, resultadosFinales); (

  if (contResultados >10) { ( 1 UT )
     maxOcurrencias = maxOcurrencias(resultadosFinales); ( 5 UT)
     s(semMaxOcurrencias)
     finalizar(); (1 UT)
   }
}
```

La planificación cambiaria dado que el acumulador sólo ejecutaría si hay resultados, evitando la excepción. A su vez, el notificador se quedará esperando que se actualice maxOcurrencias, notificando así el valor correcto.

2) 16 segmentos -> 4 bits para direccionarlos (1 hexa) -> 12 bits restantes para offset (3 hexas)

a) 20FCh -> Segmento 2, Offset 0FCh -> 0FC < FEh -> Dir fisica = FA00h + 0FCh = FAFCh
1A33h -> Segmento 1, Offset A33h -> A33h > 1Ah -> Acceso invalido

b) 1ACCh -> Solo puede pertenecer al segmento 0. Offset = 1ACCh - 1000h = ACCh -> Dir Logica = 0ACCh
70A1h -> No pertenece a ningun segmento de este proceso.

c) MV en un esquema de segmentación es difícil de implementar, dado que los segmentos tienen tamaños variables. Esto hará que la particion de swap tenga problemas de compactación y su acceso sea lento. Una solución intermedia es usar segmentación paginada.

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 comenzara más tarde sólo podrá utilizar el tiempo remanente. Utilice hojas separadas para la teoría / ejercicios.