UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela Universitaria de Informática
Facultad de Informática

Calificación
1
2
3
4

SISTEMAS OPERATIVOS

Convocatoria de junio, 19 de junio de 2008 – PRIMERA PARTE

Titulación

Nombre

*** S O L U C I O N E S ***	

Dispone de dos horas para completar el examen

---

1 (6 puntos) **Test.** Marque las opciones correctas de cada apartado. En caso de que existan varias opciones ciertas, se considerará como correcta la más completa o precisa. Las preguntas no contestadas no puntúan; las mal contestadas puntúan negativamente restando un tercio de su valor. Marque la opción correcta rodeándola con un círculo. Si se equivoca, tache la respuesta equivocada y rodee la opción correcta. Escriba con tinta. Las preguntas respondidas con lápiz o con varios círculos no tachados se considerarán no contestadas.

1. ¿Cuál de estas políticas es más aconsejable para implementar un sistema de tiempo real crítico?
   a) FCFS
   b) Round Robin
   c) *Prioridades expulsivo
   d) Prioridades no expulsivo
2. Tenemos varios procesos que intentan acceder a secciones críticas sobre una zona de datos, y observamos que el algoritmo sólo funciona bien cuando hay tres o más procesos interesados en entrar en su sección crítica al mismo tiempo: si sólo hay dos procesos intentando entrar en la sección crítica, el algoritmo los deja esperando hasta que se incorpora un tercero. En este caso, este algoritmo es:
   a) válido, suponiendo que aparte de esta incidencia, cumple con la exclusión mutua, progreso y espera indefinida
   b) *inválido, porque no cumple la condición de progreso
   c) inválido, porque no cumple la condición de espera indefinida
   d) inválido, porque no cumple la condición de exclusión mutua
3. Cuando existen varios procesos en espera por un semáforo y éste queda disponible, ¿cuál de los procesos en espera se desbloquea?
   a) el que lleva más tiempo esperando
   b) el que tenga más prioridad
   c) el que tenga más prioridad, y en caso de empate, el que lleve más tiempo esperando de los empatados
   d) *depende de la implementación del semáforo
4. Una interrupción se puede atender...
   a) sólo si cuando la interrupción ocurre, el procesador se encuentra en modo privilegiado
   b) sólo si cuando la interrupción ocurre, el procesador se encuentra en modo usuario
   c) *tanto si el procesador se encuentra en modo privilegiado como en modo usuario, pero al atenderla, el procesador conmuta a modo privilegiado
   d) ninguna de las anteriores es cierta
5. ¿Cuál de estas políticas consigue un mejor tiempo medio de espera?
   a) FCFS
   b) *SJF (expulsivo o no)
   c) Round Robin
   d) Multicola FCFS + Round Robin
6. Suponga un computador que va instalado dentro de una sonda espacial destinada a explorar la superficie de Marte. Este computador se encarga del desplazamiento de la sonda, ejecutando tareas tales como accionar las ruedas, detener la sonda si el nivel eléctrico es bajo y reaccionar ante choques con obstáculos imprevistos. ¿Qué tipo de sistema operativo sería más adecuado para este computador?
   a) un sistema por lotes
   b) un sistema de tiempo compartido
   c) *un sistema de tiempo real
   d) un sistema distribuido
7. Un sistema multiusuario:
   a) debe tener la capacidad para la ejecución concurrente de varios procesos
   b) permite que varios usuarios trabajen al mismo tiempo sobre él

c) debe repartir la memoria principal entre los distintos usuarios
d) *ninguna de las afirmaciones es correcta en todos los casos

8. ¿Cuál de estas llamadas al sistema considera usted más esencial en un sistema operativo no multiprogramado?
   a) crear un semáforo
   b) solicitar un bloque de memoria libre
   c) regresar a la cola de preparados (Yield)
   d) *cargar y ejecutar un programa
9. A un planificador de procesos que tiene la cola de preparados vacía llegan al mismo tiempo seis procesos, todos de duración 2 u.t. Si no ocurren más eventos externos mientras se atiende a estos seis procesos, ¿cuál será el tiempo medio de espera para este conjunto de procesos?
   a) 5 u.t. para FCFS
   b) 5 u.t. para FCFS y SJF
   c) *5 u.t. para FCFS, SJF y RR con cuanto igual a 2 u.t.
   d) 5 u.t. para cualquier algoritmo de planificación
10. Para el caso del apartado anterior, suponga que le añadimos la restricción de que existe un planificador de largo plazo que no permite que existan más de tres procesos ejecutándose en el planificador de corto plazo. ¿Variaría en algo la respuesta anterior?
    a) *no, la respuesta correcta seguiría siendo la misma
    b) la respuesta correcta sería diferente, ya que el tiempo medio de espera sería siempre mayor que 5 u.t. para cualquier algoritmo
    c) la respuesta correcta sería diferente, ya que el RR con cuanto igual a 2 u.t. daría un tiempo de espera mayor
    d) la respuesta correcta sería diferente, ya que el orden de ejecución de los procesos variaría en los algoritmos planteados en el anterior apartado
11. Gracias a la utilización de máquinas virtuales,
    a) podemos hacer que el núcleo de nuestro sistema operativo sea más eficiente, ya que se utilizan menos pasos intermedios para resolver una llamada al sistema
    b) *podemos hacer coexistir simultáneamente varios sistemas operativos sobre el mismo hardware
    c) podemos construir sistemas operativos más resistentes a los fallos del hardware, especialmente de los periféricos de E/S
    d) todas las afirmaciones anteriores son ciertas
12. ¿Cuál de las siguientes afirmaciones es correcta?
    a) El modo dual de operación se ha diseñado para que los programas de usuario obtengan acceso pleno a todos los recursos del sistema con total libertad.
    b) *Cuando se produce una interrupción, la CPU conmuta a modo privilegiado/supervisor.
    c) Las instrucciones de desactivación de interrupciones se ejecutan normalmente en modo de usuario.
    d) Las instrucciones lectura del reloj del sistema normalmente deberían ser operaciones privilegiadas.
13. ¿Cuáles de estas parejas de algoritmos no se pueden combinar para construir un sistema multicola?
    a) FCFS con Round-Robin
    b) Un algoritmo expulsivo con un algoritmo basado en prioridades
    c) Round-Robin con un algoritmo basado en prioridades
    d) *todas las anteriores combinaciones se pueden utilizar
14. ¿Qué módulo del sistema operativo se encarga habitualmente de atender las llamadas al sistema?
    a) la API
    b) el *dispatcher* o despachador
    c) el intérprete de órdenes o *shell*
    d) *el núcleo
15. Si un proceso ejecuta una operación P sobre un semáforo binario,
    a) el proceso se bloquea hasta que otro proceso invoque una operación P
    b) el proceso se bloquea hasta que otro proceso invoque una operación V
    c) *el proceso se bloquea solamente si el semáforo tiene valor cero
    d) el proceso se bloquea solamente si el semáforo tiene valor uno

---

2 (2 puntos) En relación con las soluciones al problema de la sección crítica:

¿Es posible que exista un algoritmo que cumpla el requisito de exclusión mutua, pero que no cumpla el requisito de progreso?

El requisito de exclusión mutua se puede cumplir perfectamente con un algoritmo que no permita entrar NUNCA a ningún proceso en su sección crítica. Como nunca hay nadie en sección crítica, siempre se verifica que no hay más de un proceso en

Nombre

ella. Pero ese algoritmo trivial no cumple el requisito de progreso, ya que, obviamente, el sistema nunca toma una decisión de dejar entrar a nadie.

Hay otros algoritmos menos triviales que sólo cumplen el requisito de exclusión mutua. Uno que hemos presentado en la asignatura es la solución para dos procesos basada en la variable «turno», que va alternando el derecho de entrar en sección crítica entre un proceso y otro. En esta solución, un proceso que quiere entrar en su sección crítica y que no tenga el turno debe esperar a que su compañero entre en la sección crítica; si el compañero no está interesado, no se cumple el requisito de progreso, ya que se está impidiendo a un proceso entrar en sección crítica aunque no haya conflicto.

¿Y es posible que exista un algoritmo que cumpla los requisitos de exclusión mutua y de progreso, pero que no cumpla el de espera limitada?

Un algoritmo de sección crítica que cumpla exclusión mutua y progreso, pero no espera indefinida puede ser la solución básica con instrucciones atómicas (test-and-set o swap). Este es el algoritmo básico con test-and-set:

```
while test-and-set (cerrojo) { NADA; }
... sección crítica ...
cerrojo = false;
```

El algoritmo consigue exclusión mutua: si varios procesos intentan acceder al mismo tiempo al cerrojo, sólo uno de ellos consigue ponerlo a «true», debido a que «test-and-set» se ejecuta de forma indivisible. Además, si uno o varios procesos intentan entrar en sección crítica estando libre, sabemos que uno de ellos acabará saliendo del bucle «while», así que se cumple el requisito de progreso. Sin embargo, este algoritmo no garantiza una espera limitada, ya que en caso de que varios procesos compitan por la sección crítica, el ganador puede ser uno cualquiera de ellos, sin que se respete la antigüedad de los contendientes o el número de veces que han conseguido entrar en el pasado. O sea que un proceso puede verse postergado indefinidamente y no entrar jamás en sección crítica, situación improbable en la vida real pero teóricamente posible, y que por tanto incumple el requisito de espera limitada.

Otro algoritmo que no cumple la condición de espera limitada es la solución básica con semáforos, en determinadas circunstancias:

```
P (cerrojo)
... sección crítica ...
V(cerrojo)
```

Esta solución funciona bien a grandes rasgos, pero el cumplimiento de la espera limitada depende de cómo esté implementado el semáforo y cuál sea la política de gestión de los procesos que esperan en la «P». Si el semáforo se gestiona como una cola FIFO no hay problema. Pero si el semáforo se gestiona como una cola de prioridades, no podemos garantizar espera limitada de un proceso con muy baja prioridad. Ocurre lo mismo si la operación «P» se implementa con un bucle de espera activa (*spinlock*), ya que en caso de espera simultánea de varios procesos, no podemos asegurar quién se libera cuando la sección crítica queda libre.

¿Por qué la solución consistente en desactivar las interrupciones mientras dura la sección crítica no se considera una solución universalmente aceptable?

Porque es una solución que depende de la existencia de un sistema centralizado de control de interrupciones, que no está disponible en todas las arquitecturas. En particular, esta solución no es aplicable en muchos entornos multiprocesadores, en los que cada procesador dispone de su propia línea de interrupciones. Cada procesador puede desactivar sus propias interrupciones, pero no puede hacerlo simultáneamente en todos los otros procesadores. Como la desactivación generalizada de las interrupciones en todos los procesadores no será atómica, no sirve de base para construir una solución a la sección crítica.

---

3 (1 punto) Para planificar procesos en un sistema multiprocesador podemos implementar las colas de preparados de dos formas: o bien utilizamos una única cola de preparados compartida por todos los procesadores; o bien hacemos que cada procesador disponga de su propia cola de preparados. ¿Qué ventajas e inconvenientes tiene cada una de las dos técnicas?

Si empleamos una única cola de preparados, tenemos el inconveniente de que debemos tratar adecuadamente el acceso simultáneo a dicha cola por parte de los procesos del sistema. Para evitar que la cola se corrompa, debemos garantizar exclusión mutua en las operaciones más básicas, como la inserción y extracción. Por el contrario, si cada procesador trabaja con su propia cola, no hay conflictos de uso.

Si cada procesador tiene su propia cola, el problema del acceso simultáneo ya no existe, pero nos surge un inconveniente más complicado de tratar: el desequilibrio de la carga de trabajo. Un procesador puede tener su cola completamente vacía y al mismo tiempo otro procesador puede estar saturado de trabajo. Si queremos aprovechar al máximo los procesadores, estamos obligados a transportar procesos de un procesador a otro si alguno de ellos queda ocioso. Este problema no existe si sólo hay una cola, ya que cuando un procesador queda libre, sólo tiene que acudir a la cola para dar servicio a otro proceso.

---

4 (1 punto) Suponga que tenemos un sistema con una cantidad de procesadores infinita. ¿Seguiría teniendo sentido que el sistema operativo se preocupe por gestionar este recurso de capacidad infinita? ¿Qué algoritmos de planificación aplicaría usted en este sistema?

Si el número de procesadores fuera ilimitado, no tendríamos que preocuparnos por repartir cada procesador entre varios procesos, ni aprovechar el tiempo ocioso de un procesador cuando el proceso actual se bloquea. Es decir, cada proceso podría tener asignado un procesador para él en exclusiva, sin límites de tiempo, prioridades respecto a sus compañeros, etc. El algoritmo de planificación para este sistema puede ser tan sencillo como dejar ejecutarse de inmediato a cualquier proceso que pase al estado de preparado. El procesador se le puede asignar en el arranque del proceso. No haría falta ni siquiera llevar una lista de procesadores libres u ocupados: tan sólo haría falta numerarlos y concederlos a los procesos en orden ascendente.

En el supuesto sólo nos han mencionado que los procesadores son infinitos, pero no se nos indica nada al respecto de otros recursos, como por ejemplo la memoria. Si hubiera recursos escasos, sí que tendría sentido aplicar políticas convencionales de planificación, para evitar que se sature alguno de esos recursos. Por ejemplo, si la memoria tiene un tamaño limitado, el número de procesos que pueden estar presentes también es limitado y por tanto, cuando llega un nuevo proceso, tiene sentido aplicar una política que limite su ejecución en función de parámetros como su prioridad, consumo estimado de memoria, etc.