



Universidad de Las Palmas de Gran Canaria  
Escuela Universitaria de Informática  
Facultad de Informática

Sistemas Operativos  
Examen Parcial, 3 de mayo de 2003

Calificación
1
2
3
4
5

## SOLUCIONES

Nombre

Titulación

Todas las preguntas puntúan lo mismo

Dispone de dos horas y media para realizar el examen

1 Responda con brevedad y precisión (5 líneas máximo) a las siguientes cuestiones:

- a) ¿Por qué el uso de la técnica de *spooling* motivó el desarrollo de la técnica de multiprogramación?

Porque desarrolló el concepto de lista de tareas (pool de Job) que podía ser gestionada de acuerdo a políticas distintas a la FIFO.

- b) ¿Con qué objetivo básico de los sistemas operativos tiene que ver el uso de las llamadas al sistema? Justifique su respuesta.

Con el de la protección, ya que los recursos son manejados en última instancia por el sistema operativo para evitar que los programas de usuario los manejen directamente.

- c) Desde el punto de vista del sistema operativo, ¿qué beneficio aporta el uso de interrupciones?

c) Que el tratamiento de eventos se realice cuando éstos se producen realmente y así no incurrir en el fenómeno de espera activa. De esta manera se hace un uso más eficiente del procesador.

- d) ¿Qué ventajas tiene el utilizar hilos como elemento básico de ejecución frente a los procesos?

Que son entidades más fáciles de manejar, ya que consumen menos recursos que los procesos. Así, el cambio de contexto entre hilos de un mismo proceso pesado es menos costoso. Por otro lado, la comunicación entre hilos es más sencilla (usando la memoria que comparten).

2 Dada la siguiente carga de trabajo, obtener el diagrama de *Gantt*, el tiempo medio de retorno y el tiempo medio de espera al aplicar las siguientes políticas de planificación:

- Primero el más corto expulsivo (SRTF).
- Round-Robin con cuanto de 4 unidades de tiempo

Proceso	Tiempo de llegada	Duración de ráfaga de CPU
P0	0	8
P1	1	4
P2	3	2
P3	5	3

**Primero el más corto expulsivo (SRTF) (admite dos posibilidades, ya que cuando llega P2 tenemos a P1 y P2 con dos ráfagas pendientes)**

#### SOLUCION A

P0	P1	P1	P1	P1	P2	P2	P3	P3	P3	P0	P0	P0	P0	P0	P0	P0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Proceso	Tiempo de espera	Tiempo de retorno
P0	9	17
P1	0	4
P2	2	4
P3	2	5
<b>Media</b>	<b>3,25</b>	<b>7,5</b>

#### SOLUCION B

P0	P1	P1	P2	P2	P1	P1	P3	P3	P3	P0	P0	P0	P0	P0	P0	P0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Proceso	Tiempo de espera	Tiempo de retorno
P0	9	17
P1	2	6
P2	0	2
P3	2	5
<b>Media</b>	<b>3,25</b>	<b>7,5</b>

#### Round-Robin (Q = 4 u.t.)

P0	P0	P0	P0	P1	P1	P1	P1	P2	P2	P0	P0	P0	P0	P3	P3	P3
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Proceso	Tiempo de espera	Tiempo de retorno
P0	6	14
P1	3	7
P2	5	7
P3	9	12
<b>Media</b>	<b>5,75</b>	<b>10</b>

**3** En una tienda de pájaros están teniendo problemas para tener a todos sus canarios felices. Los canarios comparten una jaula en la que hay un plato con alpiste y un columpio para hacer ejercicio. Todos los canarios quieren inicialmente comer del plato y, después columpiarse. Pero se encuentran con el inconveniente de que sólo tres de ellos pueden comer del plato al mismo tiempo y sólo uno puede columpiarse. Define un proceso que ejecuten los canarios concurrentemente de forma que sincronicen sus actividades usando semáforos.

```
// Semáforos compartidos por todos los canarios
puedecomer: semáforo;
columpio: semáforo;

//Inicialización de los semáforos
inicializa(puedecomer,3);
inicializa(columpio,1);

// Código que ejecutaría cada canario

// Otras operaciones
espera(puedecomer);
// Comer
señal(puedecomer);
espera(columpio);
//montar en el columpio
señal(columpio);
//otras operaciones
```

**4** A continuación se propone una solución al problema de la sección crítica para dos procesos ¿es correcta la propuesta?. Justifique su respuesta.

**Repeat**

```
flag[i] := true;
while ( flag[j] ) do
  Begin
    flag[i] := false;
    While ( flag[j] ) do no-op;
    flag[i] := true;
  End;
```

Sección crítica

```
flag[i] := false;
```

Sección no crítica

**Until** false;

**NOTA:** Los procesos están numerados como  $P_0$  y  $P_1$ . Cuando se estudie el comportamiento del proceso  $P_i$ , entonces el otro proceso será  $P_j$ ; es decir si  $i = 0$  entonces  $j = 1$ , o si  $i = 1$  entonces  $j = 0$ .

Para comprobar que el algoritmo es correcto, hay que demostrar que cumple las tres condiciones de exclusión mutua, progreso y espera limitada.

La condición de exclusión mutua se cumple. Mientras un proceso está en sección crítica, su "flag" está a True, con lo cual impide al otro proceso superar el bucle "while". Si ambos procesos pretenden entrar al mismo tiempo, puede verse que al menos uno de los dos quedará detenido en el bucle "while".

La condición de progreso también no se cumple. Es cierto que si un proceso no desea entrar en sección crítica, su "flag" está a False, no impide al otro proceso entrar, si éste lo desea (no entra en el bucle "while"). Sin embargo, si ambos desean entrar, para que se cumpla la propiedad de progreso la decisión de quien entra debe depender exclusivamente de ellos (y así ocurre) Y LA SELECCIÓN NO DEBE POSTERGARSE INDEFINIDAMENTE (esta parte es la que no se garantiza). Podemos verlo si ambos procesos intentan entrar en la sección crítica a la vez y avanzan al mismo ritmo. Ambos pondrán su "flag" a True, con lo cual los dos entrarán en el bucle "while". En cada iteración, los dos procesos colocan temporalmente su "flag" a falso, superan el "while" interior, vuelven a poner su "flag" a True y entran en otra iteración. Esta situación puede repetirse indefinidamente, no hay garantía formal de que el número de iteraciones esté acotado.

Aunque la comprobación de la condición anterior deja claro que la solución no es correcta, podemos añadir que tampoco se cumple la condición de espera limitada. Se puede comprobar fácilmente que existe la posibilidad de que el proceso  $P_1$  se quede indefinidamente en la sección de entrada a la sección crítica debido a que el proceso  $P_0$  entre repetidamente en su sección crítica.

**5 Test.** En cada uno de los siguientes apartados, señale cuál opción es correcta. En caso de que existan varias opciones ciertas, se considerará como correcta la más completa o precisa. Las preguntas no contestadas no puntúan; las contestadas erróneamente penalizan un tercio de su valor. Señale sus respuestas rodeando con un círculo la opción correcta. Si se equivoca, tache la respuesta incorrecta y rodee la opción que considere correcta. Escriba con tinta. Las preguntas respondidas con lápiz o con varios círculos no tachados se considerarán no contestadas.

**1. La planificación de procesos de nivel intermedio está estrechamente relacionada con:**

- a. La planificación del procesador
- b. La planificación de las operaciones de Entrada/Salida
- c. La planificación de memoria**
- d. La planificación del espacio en disco

**2. En un sistema de tiempo real es crítico:**

- a. El grado de ocupación del procesador
- b. El flujo de trabajo
- c. El tiempo de espera
- d. El tiempo de respuesta**

3. La confiabilidad es una propiedad característica de los sistemas:
- a. Multiprogramados
  - b. De tiempo compartido
  - c. De tiempo real
  - d. Distribuidos**
4. El sistema operativo OS/2 es un caso de sistema operativo con modelo:
- a. Monolítico
  - b. Por capas**
  - c. Máquina virtual
  - d. Cliente/Servidor
5. Si una solución al problema de la sección crítica da lugar a inanición, entonces no cumple la condición:
- a. Exclusión mutua
  - b. Espera limitada**
  - c. Progreso
  - d. Todas las anteriores son falsas
6. ¿El problema de la sección crítica puede ser resuelto con semáforos?
- a. Siempre
  - b. Depende de cómo estén implementados los semáforos**
  - c. Nunca
  - d. Todas las anteriores son falsas
7. El efecto convoy se puede producir en sistemas con planificación del procesador:
- a. FIFO**
  - b. SJF no expulsivo
  - c. SJF expulsivo
  - d. Round-Robin
8. El fenómeno de inanición no se puede producir en sistema con planificación del procesador:
- a. FIFO
  - b. SJF no expulsivo
  - c. SJF expulsivo
  - d. Round-Robin**
9. El intérprete de órdenes,
- a. es un programa del sistema
  - b. Obtiene y ejecuta las órdenes especificadas por el usuario
  - c. hace uso de las llamadas al sistema para desempeñar su función
  - d. Todas las anteriores son ciertas**

**10. No tiene sentido hablar de**

- a. un sistema multiprogramado monousuario
- b. un sistema de tiempo compartido multiusuario
- c. un sistema multiprogramado y multiprocesador
- d. un sistema de tiempo compartido no multiprogramado**

**11. El que ciertas instrucciones se llamen "atómicas" significa que:**

- a. no se pueden ejecutar concurrentemente
- b. se han de ejecutar indivisiblemente**
- c. No tiene ningún significado especial
- d. Todas son falsas

**12. Se ejecutan consecutivamente dos operaciones *P* o *WAIT* sobre un semáforo. ¿ Es esto posible ?**

- a. Sí**
- b. Sí, siempre que no hayan sido procesos diferentes
- c. No
- d. No, si las ha realizado un solo proceso

**13. La utilización de una cola de procesos para la implementación de las operaciones con semáforos:**

- a. es necesaria para poder realizar una implementación de los mismos
- b. evita las soluciones con espera activa**
- c. es posible gracias a las instrucciones atómicas (*test-and-set* y *swap*)
- d. todas las anteriores son ciertas

**14. Un sistema monoprocesador que lleve a cabo una planificación no expulsiva:**

- a. No es implementable
- b. No puede ser multiprogramado ya que para ello es necesario una política expulsiva
- c. Presenta menos dificultades para controlar el problema de la sección crítica**
- d. todas las anteriores son falsas

**15. Sobre planificación de procesos:**

- a. Todos los métodos basados en prioridades tienen riesgo de inanición
- b. Los métodos multicolos están concebidos para planificar sistemas multiprocesadores
- c. El cuanto de tiempo óptimo para un *Round Robin* viene influenciado por el tiempo de cambio de contexto
- d. a) y c) son ciertas**