

Examen de Programación Concurrente

Junio 2007
Departamento de Lenguajes, Sistemas Informáticos e Ingeniería del Software

Normas

Este examen es un cuestionario tipo test que consta de **5 preguntas** en **2 páginas**. La puntuación total del examen es de **10 puntos**. La duración total es de **una hora**. El examen debe contestarse en las **mismas hojas**. No olvidéis rellenar **apellidos, nombre y número de matrícula** en cada hoja.

Sólo hay una respuesta válida por pregunta. Toda pregunta en que se marque más de una respuesta se considerará incorrectamente contestada. Toda pregunta incorrectamente contestada restará del examen una cantidad de puntos igual a la puntuación de la pregunta dividido por el número de alternativas ofrecidas en la misma.

La solución al examen se proporcionará antes de la revisión. Las calificaciones se darán a conocer el **18 de junio**. La revisión del examen tendrá lugar el **20 de junio**.

Cuestionario

- (2 puntos) 1. Con el código de la **figura 1** un programador desea utilizar dos semáforos binarios S y T para *asegurar* más la exclusión mutua en una sección crítica:

Suponiendo que dos procesos intenten ejecutar el código anterior y comparten todas las variables que pudieran aparecer en él, seleccionar la respuesta correcta:

- (a) No hay ningún problema. Se consigue exclusión mutua y no hay interbloqueos.
- (b) No va a haber ningún interbloqueo, pero no se garantiza la exclusión mutua.
- (c) Se garantiza la exclusión mutua pero puede haber interbloqueos.
- (d) No se garantiza la exclusión mutua y es posible que haya interbloqueos.

```

1  loop
2      Wait (S) ;
3      Wait (T) ;
4          -- Sección crítica
5      Signal (S) ;
6      Signal (T) ;
7  end loop;
```

Figura 1: Sección crítica *reforzada* — sólo para la **pregunta 1**.

```

1  loop
2      Wait (S) ;
3      Wait (T) ;
4          -- Sección crítica
5      Signal (T) ;
6      Signal (S) ;
7  end loop;
```

Figura 2: Sección crítica *reforzada* — sólo para la **pregunta 2**.

- (2 puntos) 2. En la **figura 2** un programador desea utilizar dos semáforos binarios S y T para *asegurar* más la exclusión mutua en una sección crítica:

Suponiendo que dos procesos intenten ejecutar el código anterior y comparten todas las variables que pudieran aparecer en él, seleccionar la respuesta correcta:

- (a) No hay ningún problema. Se consigue exclusión mutua y no hay interbloqueos.
- (b) No va a haber ningún interbloqueo, pero no se garantiza la exclusión mutua.
- (c) Se garantiza la exclusión mutua puede haber interbloqueos.
- (d) No se garantiza la exclusión mutua y es posible que haya interbloqueos.

- (2 puntos) 3. Se quiere implementar un procedimiento / método $P()$, cuyas condiciones de concurrencia dependen de los parámetros de entrada, con el código que aparece en la figura 3(a) y (b). Decir cuál de las siguientes afirmaciones es correcta:

- (a) No va a compilar,¹ porque en la figura 3(b) se está utilizando el parámetro J como índice y debería llamarse I , como en el **requeue** de la figura 3(a).

¹Exceptuando, por supuesto, errores tipográficos y similares. Consultar con el profesor en caso de duda.

(a) Externa	(b) Aplazada versión 1	(c) Aplazada versión 2
<pre> entry P(X: T) when True is I: Natural := 0; begin while not V(I).Ocup loop I := I + 1; end loop; V(I).Ocup := True; V(I).Args := X; V(I).PID := I; requeue P_Ap(I); end P; </pre>	<pre> entry P_Ap (for J in Tipo_PID) (X: T) when Q(J) is begin V(J).Ocup := False; ... end P_Ap; </pre>	<pre> entry P_Ap (for J in Tipo_PID) (X: T) when Q(J) is K: Natural := 0; begin while V(K).PID /= J loop K := K + 1; end loop; V(K).Ocup := False; ... end P_Ap; </pre>

Figura 3: Código de ejemplo. Se supone que el vector $V()$ ha sido convenientemente dimensionado e inicializado.

- (b) Es correcto. No hay ningún problema si las condiciones están bien implementadas.
- (c) Puede fallar en tiempo de ejecución: en la figura 3(b) no se busca qué J corresponde a la llamada en curso; es decir, se está utilizando sin haber sido correctamente inicializado.
- (d) Se está ejecutando $V(J).Ocup := False$ demasiado pronto: hay que esperar a que se acabe de ejecutar completamente la **entry** correspondiente. Si se hace como está en la figura 3(b) otro proceso podría *apropiarse* del componente J -ésimo y “machacar” sus contenidos.
- (2 puntos) 4. Se quiere implementar un procedimiento / método $P()$, cuyas condiciones de concurrencia dependen de los parámetros de entrada, con el código que aparece en la figura 3(a) y (c). Decir cuál de las siguientes afirmaciones es correcta:
- (a) Falta la condición de terminación del bucle. Si no hay ningún elemento del vector con $V(K).PID = J$ va a intentar acceder a elementos que no están en el vector y terminará con un error.
- (b) Es correcto. No hay ningún problema si las condiciones de concurrencia están bien implementadas.
- (c) La versión (c) puede detenerse en un elemento erróneo del vector: el índice k en el que se para la búsqueda puede tener un $pid = j$ que haya sido almacenado en una llamada anterior o en la inicialización de las variables. Los datos que estén almacenados pueden no tener nada que ver con los correspondientes a la llamada que nos ocupa.
- (d) La línea $V(K).Ocup := False$ debería ser $V(J).Ocup := False$. No funciona si no es así.
- (2 puntos) 5. Se ha implementado la especificación de la izquierda con el código de la derecha, usando objetos protegidos:

CPRE: $Q(r) \wedge S(r, x)$
 $P(r, x)$
POST: ...

```

entry P(X: T)
  when Q is
begin
  while not S(X) loop
    null;
  end loop;
  -- La implementación de la postcondición va aquí
end P;

```

Como siempre, r se refiere al estado del recurso y x es un parámetro de entrada. Tanto Q como $S(X)$ pueden acceder al estado del recurso y $S(X)$ implementa únicamente la parte de la precondición $S(r, x)$. ¿Cuál de las siguientes respuestas es correcta?

- (a) Es incorrecto. No implementa la especificación.
- (b) Es correcto, pero realiza espera activa, por lo que no es buena idea.
- (c) Ni siquiera va a compilar.
- (d) Siempre se va a bloquear al ejecutarse.