

SCDTestTeoria.pdf



Anónimo



Sistemas Concurrentes y Distribuidos



2º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación
Universidad de Granada

BBVA

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

Ventajas Cuenta Online de BBVA

0€
Sin comisión de administración o mantenimiento de cuenta.
(0 % TIN 0 % TAE)

0€
Sin comisión por emisión y mantenimiento de Tarjeta
Aqua débito.

0
Sin necesidad de domiciliar nómina o recibos.

Ábrete la Cuenta Online de BBVA y
llévate 1 año de **Wuolah PRO**

cómo??

Las ventajas de **WUOLAH PRO**

✖ Di adiós a la publi en los apuntes y en la web

✖ Descarga carpetas completas de un tirón

✖ Acumula tickets para los sorteos





Ábrete la Cuenta Online de BBVA y
llévate 1 año de Wuolah PRO

cómo??
↗



1/6

Este número es
indicativo del
riesgo del
producto, siendo
1/6 indicativo de
menor riesgo y
6/6 de mayor
riesgo.

BBVA está
adherido al
Fondo de
Garantía de
Depósitos de
Entidades de
Crédito de
España.
La cantidad
máxima
garantizada es
de 100.000 euros
por la totalidad
de los depósitos
constituidos
en BBVA por
persona.

ventajas

PRO



Dí adiós a la publi
en los apuntes y
en la web



Acumula tickets
para los sorteos



Descarga
carpetas
completas

estudia sin publi
WUOLAH PRO

TEST TEORÍA

**NO PONGAIS PANTALLAZOS PORQUE EL DIA DEL EXAMEN
NO HAY TIEMPO DE VER SI ESTÁ AQUÍ O NO.**

Considera dos procesos que acceden concurrentemente a una variable entera compartida x, inicializada a 0, según se indica aquí

```
process A ;           process B ;
begin                  begin
  x := x+1 ;          x := x+3 ;
end                   print(x) ;
                      end
```

El valor impreso por el programa

- puede ser 1,2,3 o 4
- **puede ser 1, 3 o 4**
- puede ser 3 o 4
- siempre será 3

Considera dos procesos que acceden concurrente a una variable entera compartida x, inicializada a 0, según se indica aquí

```
process A ;           process B ;
begin                  begin
  x := x+2 ;          x := x+4 ;
end                   print(x) ;
                      end
```

el valor impreso por el programa

- puede ser 2,3,4 5 o 6
- puede ser 4 o 6
- siempre será 4
- **puede ser 2, 4 o 6**

Considera dos procesos que accede concurrente a una variable entera compartida x, inicializada a 0, según se indica aquí

process A ; begin x := x+3 ; print(x) ; end	process B ; begin x := x+2 ; end
---	---

el valor impreso por el programa

- puede ser 3 o 5
- siempre será 3
- **puede ser 2, 3 o 5**
- puede ser 2, 3, 4, o 5

12. Dado el siguiente ejemplo de paso de mensaje entre dos procesos, señala la opción correcta

Process P0:
Var dato_env : integer;
Begin
 Dato_env := 10;
 Send (dato_env , P1);
 Dato_env := 15;
End

Process P1:
Var dato_rec : integer;
Begin
 Dato_rec := 20;
 receive (dato_rec , P2);
 imprime (dato_rec);
End

- Si el comportamiento es seguro, P1 imprimirá 20.
- **Si el comportamiento no es seguro, P1 imprimirá 15.**
- Si el comportamiento es seguro, P1 imprimirá 15.
- Si el comportamiento no es seguro, P1 sólo puede imprimir 20



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

Ábrete la Cuenta Online de BBVA y llévate 1 año de Wuolah PRO



cómo??

Las ventajas de Wuolah PRO



Di adiós a la publi en
los apuntes y en la web



Descarga carpetas
completas de un tirón



Acumula tickets
para los sorteos

Ventajas Cuenta Online de BBVA

0€

Sin comisión de administración o
mantenimiento de cuenta.
(0 % TIN 0 % TAE)

0€

Sin comisión por emisión y
mantenimiento de Tarjeta
Aqua débito.

0

Sin necesidad de domiciliar
nómina o recibos.

Suponiendo que un semáforo s tiene en un instante el valor 0, y que a partir de ese instante un proceso ejecuta estas instrucciones

sem_wait(s);

sem_signal(s);

sem_wait(s);

suponiendo que ningún otro proceso accede al semáforo en ningún momento, entonces, el proceso que ejecuta las instrucciones

- quedará bloqueado en sem_signal
- quedará bloqueado en el segundo sem_wait
- quedará bloqueado en el primer sem_wait
- ejecutará las tres instrucciones sin quedarse bloqueado en ninguna de ellas

Suponiendo que un semáforo s tiene en un instante el valor 0, y que a partir de ese instante un proceso ejecuta estas instrucciones

sem_signal(s);

sem_wait(s);

sem_wait(s);

suponiendo que ningún otro proceso accede al semáforo en ningún momento, entonces, el proceso que ejecuta las instrucciones

- quedará bloqueado en sem_signal
- ejecutará las tres instrucciones sin quedarse bloqueado en ninguna de ellas
- quedará bloqueado en el segundo sem_wait
- quedará bloqueado en el primer sem_wait

1. Un sistema de tiempo real:

- Se considera que es incorrecto si es menos fiable que un sistema convencional
- Se considera que es incorrecto si no se cumple la corrección funcional
-  - Solo se considera incorrecto si no se cumplen ni la corrección temporal ni la corrección funcional.
- Solo se considera incorrecto si no se cumple la corrección temporal.

2. Supongamos que un proceso P1 inicia la ejecución de A; send(v,P2); al tiempo que P2 inicia la ejecución de i_receive(v,P1);B; (utilizando recepción insegura). Entonces:

- La sentencia A no inicia su ejecución antes del fin de B
- No se puede decir nada del orden relativo de A y B
- La sentencia A termina de ejecutarse después del inicio de la sentencia B
- La sentencia B no inicia su ejecución antes del fin de A

3. Si dos procesos quieren tener una cita mediante el envío y recepción de un único mensaje, debe usar:

- Envío síncrono (s_send) y recepción síncrona(receive)
- Variables compartidas entre el proceso emisor y el receptor
- Envío síncrono (s_send) y recepción insegura (i_receive)
- Cualquier combinación de modalidades de envío/recepción constituye una cita

4. En un sistema de tiempo real en el que se dispone de un procesador y hay que ejecutar tres tareas periódicas t_1, t_2, t_3 , con tiempo de ejecución (C) 3, 4 y 6 y periodos (T) 6, 12 y 12, respectivamente. Llamamos T_m a la duración del ciclo principal. Bajo planificación cíclica:

- No sería planificable
- $T_m = 72$
- $T_m = 24$
- $T_m = 12$

5. En un sistema de tiempo real en el que se dispone de un procesador y hay que ejecutar tres tareas periódicas t_1, t_2 y t_3 , todas con periodo (T) 100 y tiempo de ejecución (C) 65, 75 y 65, respectivamente. Bajo planificación RMS.

- Si no pasara el test de Liu & Layland, habría que hacer un cronograma para determinar si es planificable.
- Si hubiera dos procesadores, sería planificable por que nos alcanza el 200% de utilización.

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publ en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publ

WUOLAH PRO

- Seguro que el sistema no es planificable.
- Si no pasara el test de Liu & Layland, no sería planificable.

6. Respecto a los conceptos básicos sobre sistemas de paso de mensajes:

- En un sistema basado en memoria compartida es imposible implementar un sistema de paso de mensajes.
- El proceso emisor y el receptor deben compartir, como mínimo, una variable.
- Cualquier modalidad de envío/recepción permite transferir datos, algunas, además, permiten sincronización.
- Las primitivas básicas de paso de mensajes permiten transferir datos, pero no sincronización.

TEMAS 3 Y 4

7. Seleccionar la única respuesta correcta respecto de la orden MPI_Receive

- El proceso que programe dicha orden siempre se bloquea independientemente del estado del búfer (vacío o con datos esperando)
- Sólo si suponemos un mecanismo de comunicación síncrono sin búfer ("citas") un proceso que programe dicha orden siempre se bloqueará
- Si la ejecución de la orden anterior no bloquea al proceso, entonces no podemos asegurar que la operación de transmisión de los datos sea segura
- ✓ - El proceso que programe dicha orden se bloqueará sólo si el búfer se encuentra vacío, es decir, no hay mensajes pendientes de ser recibidos

8. Seleccionar la única respuesta correcta en el caso de que un proceso programe la operación insegura MPI_Irecv(...):

- La reducción del tiempo de recepción del mensaje en el receptor es independiente de que el sistema de ejecución cuente con hardware especializado o no
- ✓ - Los datos ya transmitidos siempre se mantienen en el búfer del sistema hasta que el proceso receptor pueda descargarlos a su espacio de memoria y esto es independiente de que el proceso que ejecute la orden vuelva inmediatamente (sistema con hardware especializado)
- Con esta operación no se iniciará la transmisión de datos entre el proceso emisor y el receptor inmediatamente
- Con esta operación siempre se anulará el tiempo de espera en el proceso receptor

9. Indicar cuál de las siguientes afirmaciones puede considerarse correcta respecto de la Sólo de que exista algún proceso del programa que haya iniciado su envío orden de espera selectiva:



- Una vez se ha intentando seleccionar la guarda:
- Si no se ha podido, no se hace nada (no hay guardas ni ejecutables, ni potencialmente ejecutables) y finaliza la ejecución.
 - Si se ha podido, se dan estos dos pasos en secuencia:
 1. Si esa guarda tiene sentencia de entrada, se ejecuta el receive (siempre habrá un send iniciado), y se recibe el mensaje.
 2. Se ejecuta la sentencia asociada a la alternativa.A continuación finaliza la ejecución del select.

- Una vez que se ha seleccionado una orden potencialmente ejecutable, el proceso que la contiene se ejecuta secuencialmente hasta que termina el bloque componente (do...end)
- Una orden de espera selectiva no termina hasta que no ejecute cada una de sus alternativas, lo cual ocurre no determinísticamente
- La orden de espera selectiva puede programar un array de guardas indexadas, pero cada una ha de incluir una operación de entrada (receive(...))
- Una orden de espera selectiva puede entremezclar las secuencias de ejecución de las instrucciones de los bloques de sentencias componentes de cada una de sus alternativas

10. La elección y ejecución inmediata de 1 alternativa de la orden de espera selectiva (select) sólo se producirá si se cumple una de las condiciones siguientes (indicar cuál):

- Sólo depende de que exista alguna orden potencialmente ejecutable en ese momento
- Sólo depende de que alguna condición de las órdenes con guarda sea cierta
- ✓ - Existe, al menos, una orden potencialmente ejecutable y además se nombra a un proceso del programa que ya ha iniciado su envío
- Sólo de que exista algún proceso del programa que haya iniciado su envío

11. Indicar cuál de las siguientes afirmaciones es correcta con respecto a la denominada deriva acumulativa que experimentan los programas de tiempo real:
creo que no lo hemos dado

- La deriva acumulativa se produce en una tarea porque no es posible programar un retraso exacto hasta un instante en que produzca la siguiente activación, es decir, todos los retrasos que se programan indican un intervalo de suspensión relativo al instante en que se ejecutan
- Un conjunto de tareas cuya utilización del procesador es del 100% y que resulta ser planificable con el algoritmo EDF podría ser también planificable con RMS
- EDF es el único esquema de planificación dinámico
- Siempre podremos planificar con EDF cualquier conjunto de tareas cuya utilización del procesador no sea mayor del 100%, independiente de valor de su plazo de tiempo límite
-

13. Respecto a la invocación de métodos remotos (RMI) en un modelo de objetos distribuidos

- Los objetos del programa servidor que implementan la interfaz remota necesita conocer la dirección de los objetos del lado cliente
- ✓ - Los objetos del programa cliente necesitan manejar referencias remotas de los objetos del programa servidor que implementan la interfaz remota
- Los objetos del programa cliente deben crearse en el nodo servidor

- Los objetos del programa cliente deben registrarse en el enlazador (registro RMI)

14. Respeto a los sistemas de paso de mensaje con operaciones envío-recepción seguras (envío con send o s_send o con receive)

- ✓ - Si el receptor no está preparado para recibir el mensaje, el emisor podrá bloquearse o no.
- Una vez finalizada la operación de envío , siempre se puede asegurar que el receptor ha recibido el mensaje
 - El emisor y el receptor siempre se bloquean esperándose mutuamente
 - Si el mensaje no está disponible, la operación de recepción puede bloquear o no al proceso receptor.

TEMAS 1 Y 2

1. Supongamos que un algoritmo de exclusión mutua (para dos procesos) que cumplen todas las propiedades excepto la de esperas limitadas siendo ambos procesos bucles infinitos (PE + SC +RS) Entonces:
 - a. Ambos procesos pueden permanecer en PE indefinidamente
 - b. Un proceso podría quedar en PE indefinidamente mientras el otro está en RS
 - c. Puede ocurrir que un proceso permanezca en PE mientras el otro accede a SC varias veces
 - d. Todas las anteriores son falsas
2. Supongamos un algoritmo de exclusión mutua para dos procesos 1 y 2 que cumplen la propiedad de exclusión mutua y que están diseñados de forma tal que, cuando uno de ellos entra en SC, el otro (aunque este en RS) tiene garantizado que accederá a SC antes de que el primero vuelva a entrar en SC.
 - El algoritmo no cumple la propiedad de espera limitada
 - El algoritmo puede producir interbloqueo(CUANDO TODOS SE QUEDAN ESPERANDO EN PE)
 - El algoritmo no cumple la propiedad de progreso en la ejecución
 - El algoritmo es correcto
3. El programa concurrente con semáforos, en un momento determinado hay dos procesos bloqueados en el semáforo sem y otro hace un sem_signal(sem). El efecto de dicho sem_signal es:
 - Libera los dos procesos bloqueados en sem
 - Incrementa el valor del semáforo en una unidad
 - Libera a uno de los procesos bloqueados en sem
 - Libera a un proceso bloqueado en sem e incrementa el valor del semáforo en una unidad

4. En un programa concurrente que utiliza semáforos sem, que está inicializado a cero

- La primera operación sem_wait sobre el semáforo sem bloquea al proceso que lo ejecuta
- La primera operación sem_signal sobre el semáforo sem no tiene ningún efecto
- **No se puede hacer un sem_wait(sem) hasta que no se haga un sem_signal(sem).**
- Todas las anteriores son falsa

5. En un monitor ya inicializado y que está siendo utilizado por varios procesos:

- La cola del monitor puede estar vacía o tener como máximo un proceso
- **Si hay un proceso dentro del monitor, todos los demás procesos estarán esperando en la cola del monitor.**
- Si la cola del monitor está vacía, significa que hay un proceso dentro del monitor
- Todas las anteriores son falsas

6. Respecto a los semáforos, señala la opción correcta

- Si hay algún proceso en la cola de semáforo sem el valor de dicho semáforos es cero en ese momento
- Por la cola de procesos del semáforo sem pasan todos los procesos que han ejecutado un sem_wait(Sem)
- **Un proceso no puede ejecutar dos operaciones sem_wait seguidas sobre el mismo semáforo**
- Todas las anteriores son verdaderas

7. En un monitor con una semántica de señales, señalar y espera urgente, y sobre el que se han declarado variables de tipo condición

- El número total de colas de procesos que gestionan el monitor es 4
- El número total de colas de procesos que gestionan los monitores 5
- El número total de colas de procesos que gestionan el monitor es 3
- **El número total de colas de procesos que gestionan los monitores 2**

8. Respecto al concepto de monitor

- Las variables permanentes pueden ser accedidas fuera del monitor, siempre que no se modifique su valor
- **Si un proceso esta ejecutando un procedimiento del monitor(A) ningún otro proceso podrá ejecutar procedimiento A pero si otro procedimiento de dicho monitor**
- Si un proceso esta ejecutando un procedimiento del monitor (A) ningún otro proceso podrá ejecutar procedimiento A ni ningún otro procedimiento de dicho monitor
- La a y la b son las correctas

9. Respecto a los cerrojos (están basados en esperas ocupadas), señalar la opción correcta

- Los cerrojos son mecanismos que usan esperas ocupadas
- **Los cerrojos utilizan instrucciones máquina atómicas**

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publicidad en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publicidad

WUOLAH PRO

- Las instrucciones LeerAsiganr(a) esta compuesta de tres instrucciones atómicas
- La a y c son correctas

10. Respecto a un programa concurrente que utiliza un monitor

- Todas las hebras del programa deben llamar a algún procedimiento del monitor
- Lo único que las hebras del programa pueden hacer sobre el monitor es invocar a los procedimientos exportados de dicho monitor
- El monitor es un proceso que se ejecuta concurrentemente con el resto de hebras del programa
- Todas Las Anteriores Son Falsa

11. Seleccionar la única respuesta correcta, Un programa concurrente que cumpla la propiedad de seguridad para todas sus ejecuciones se considerará correcto si además:

- Todos sus procesos también cumplen la propiedad de vivacidad, lo que es equivalente a afirmar que los procesos del programa nunca pueden llegar a una situación de interbloqueo. (vivacidad, planificación, cola de monitores, política FIFO)
- Se puede demostrar que sus procesos no sufren inanición en ninguna posible ejecución del programa
- Sus procesos consiguen ejecutar sus instrucciones de forma equitativa
- Se puede demostrar que sus procesos no sufren inanición en ninguna posible ejecución del programa y además se ha de cumplir la propiedad de vivacidad

12 . Seleccionar la única respuesta correcta:

- Monitores con métodos largos (mucho código) ocasionan que la ejecución de un programa concurrente sea prácticamente secuencial, ya que sólo un proceso concurrente puede entrar al monitor cada vez
- Un proceso de un programa concurrente podría llegar a bloquearse antes de que termine la ejecución del método del monitor al que previamente ha llamado
- Los métodos de los monitores no admiten paso de parámetros por referencia porque se podrían pasar variables globales del programa como argumentos
- A diferencia de lo que ocurre con las clases en los lenguajes de programación orientados a objetos, los monitores no se pueden instanciar ya que el invariante ha de ser único

13. Seleccionar la única respuesta correcta:

- Un programa con procesos concurrentes y programado con monitores no puede implementarse en procesadores multinúcleo- sólo se puede en monoprocesadores con memoria globalmente consistente
- Como consecuencia de la llamada al procedimiento de un monitor, un proceso de un programa concurrente se puede suspender y salir del monitor

- La protección automática de las variables permanentes declaradas dentro de un monitor hace innecesaria la demostración de la propiedad de seguridad en los programas
- La planificación FIFO obligatoria de las colas de los monitores asegura en todos los casos que se satisface la propiedad de vivacidad de los procesos de un programa concurrente

14.Aunque un monitor garantiza el acceso en exclusión mutua a sus variables permanentes, los métodos de una clase-monitor han de cumplir la propiedad denominada reentrancia (un procedimiento puede ser interrumpido en medio de su ejecución y volver a ser llamado después por otros procesos del programa de manera segura). Seleccionar una de las razones siguientes para que un lenguaje con monitores exija la propiedad de reentrancia:

- Porque en los métodos de cada monitor se pueden programar varias operaciones c.wait()
- Porque durante la ejecución de un programa, los procesos concurrentes van alternándose en la entrada al monitor
- Sólo porque un determinado proceso podría entrar varias veces en un mismo monitor y tener aún pendiente la terminación de alguna de sus llamadas anteriores
- Porque se pueden anidar las llamadas a los procedimientos de un mismo monitor y un proceso podría tener pendientes de terminación más de 1 llamada

15.Sobre el significado de la semántica de las operaciones de variables condición de los monitores, seleccionar la única respuesta correcta:

- Las “señales automáticas” se consideran señales desplazantes pero no hay que demostrar la corrección de las operaciones “c.signal()”
- Con la semántica denominada “señalar y continuar” (SC) se puede producir “robo de señal” (un tercer proceso entra en el monitor antes que el proceso señalado y cambia el valor de la condición de desbloqueo) pero sólo si todos los procesos se bloquean en la misma cola
- Con semántica de “señales urgentes” puede ocurrir que la salida de un proceso del monitor provoque que un proceso suspendido en una cola del monitor vuelva a ejecutarse
- Con semántica de señales “señalar y salir” (SX) se obliga a que las operaciones c.signal() se programen necesariamente como la última instrucción de los procedimientos del monitor

Respecto al mecanismo de los cerrojos para solucionar el problema de la EM:

- Se basa en el uso de registros hardware específicos para EM
- **Se basa en el uso de instrucciones máquina atómicas**
- No utiliza espera ocupada
- No cumple la propiedad de progreso.

En un monitor con una semánticas de señales señalar y esperar (SE) y sobre el que se han declarado 4 variables de tipo condición:

- El número total de colas de procesos que gestiona el monitor es 4
- El número total de colas de procesos que gestionar el monitor es 2
- El número total de colas de procesos que gestionar el monitor es 6
- El número total de colas de procesos que gestionar el monitor es 5

La sentencia compuesta cobegin A; B coend implica que

- A y B se pueden iniciar a la vez
- A no puede acabar hasta que no acabe B
- B no puede comenzar hasta que no acabe A
- A y B se deben iniciar a la vez

Respecto a un programa concurrente que utiliza un monitor con variables condición (señales):

- Los procesos bloqueados en la cola de una variable condición solo pueden ser liberados mediante un signal a dicha variable condición
- El resultado del programa será el mismo independientemente de la semántica de las señales
- Si se ejecuta un signal sobre una variable condición, un proceso es liberado de la cola del monitor
- Los procesos que ejecutan un wait sobre una variable condición quedarán bloqueados salvo que previamente se haya hecho un signal sobre dicha variable condición

Supongamos un semáforo s inicializado a 0 al cual acceden dos procesos A y B. El proceso A ejecuta un bucle finito de 5 iteraciones, en cada iteración ejecuta sem_signal(s), y el proceso B ejecuta un bucle finito de 10 iteraciones, en cada iteración ejecuta sem_wait(s). Entonces:

- El proceso B puede quedar bloqueado de vez en cuando, pero siempre saldrá de ese bloqueo después de un tiempo finito
- Es imposible que el proceso B se bloquee en ningún momento, ni siquiera durante un tiempo
- Con seguridad el proceso B quedará bloqueado para siempre, sin poder acabar el bucle
- El proceso A puede quedar bloqueado de vez en cuando, pero siempre saldrá de ese bloqueo después de un tiempo finito.

En un monitor ya inicializado y que está siendo utilizado por varios procesos:

- Los procesos en la cola del monitor están esperando para ejecutar algún procedimiento del monitor
- Si en algún momento no hay ningún proceso dentro del monitor, el programa es erróneo

- Los procesos pueden acceder a las variables permanentes fuera del monitor, pero si es únicamente para la lectura
- La cola del monitor puede estar vacía o tener como máximo un proceso

En un instante dado durante una ejecución un programa, el estado de ejecución es

- el conjunto de variables del programa junto con sus valores en ese momento
- la secuencia ordenada de sentencias atómicas ejecutadas hasta ese instante
- la secuencia ordenada de estados producidos tras cada sentencia atómica ejecutada
- el conjunto de sentencias no atómicas de asignación a variables
-

En un monitor ya inicializado y que está siendo utilizado por varios procesos:

- La cola del monitor puede estar vacía o tener como máximo un proceso
- Si hay un proceso ejecutando código del monitor, todos los demás procesos están esperando en la cola del monitor
- Si no hay ningún proceso ejecutando código del monitor, la cola del monitor está vacía
- Si la cola del monitor está vacía significa que hay un proceso dentro del monitor

En un monitor con una semántica de señales señalar y continuar (SC) y sobre el que se han declarado 3 variables de tipo condición:

- El número total de colas de procesos que gestiona el monitor es 5
- El número total de colas de procesos que gestiona el monitor es 3
- El número total de colas de procesos que gestiona el monitor es 2
- El número total de colas de procesos que gestiona el monitor es 4

Sean A y B dos sentencias, si ejecutamos un fork A; y después de eso ejecutamos B, entonces

- A y B se pueden ejecutar a la vez, excepto en sistemas monoprocesador
- A y B se pueden ejecutar a la vez
- A y B se deben ejecutar a la vez
- B no se puede comenzar hasta que no acabe A

Sean A y B dos sentencias, si ejecutamos fork A; y después de eso ejecutamos B, entonces

- A y B se pueden ejecutar a la vez, excepto en sistemas monoprocesador
- B no puede comenzar hasta que no acabe A
- A y B se deben ejecutar a la vez

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publicidad en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publicidad

WUOLAH PRO

- A y B se pueden ejecutar a la vez

Supongamos un semáforo s inicializado a 0 al cual acceden dos procesos A y B. El proceso A ejecuta un bucle finito de 10 iteraciones, en cada iteración ejecuta sem_signal(s), y el proceso B ejecuta un bucle finito de 5 iteraciones, en cada iteración ejecuta sem_wait(s). Entonces:

- El proceso B puede quedarse bloqueado de vez en cuando, pero siempre saldrá de ese bloqueo después de un tiempo finito
- Con seguridad el proceso B quedará bloqueado para siempre, sin poder acabar el bucle
- El proceso A puede quedarse bloqueado de vez en cuando, pero siempre saldrá de ese bloqueo después de un tiempo finito
- Es imposible que el proceso B se bloquee en ningún momento, ni siquiera durante un tiempo

Dado un algoritmo de exclusión mutua para dos procesos que son bucles infinitos (PE+SC+RS), supongamos que dicho algoritmo cumple todas las propiedades salvo la de progreso; en ese caso:

- Puede ocurrir que un proceso permanezca en el PE mientras el otro accede a SC varias veces
- Puede ocurrir que ambos procesos estén en SC simultáneamente
- Los dos procesos no pueden estar simultáneamente en el PE
- Ambos procesos pueden permanecer en el PE indefinidamente

La historia o traza de un programa concurrente

Es el conjunto de variables del programa junto con sus valores en ese momento

Es la secuencia ordenada de sentencias atómicas ejecutadas hasta ese instante

Es el conjunto de sentencias no atómicas de asignación a variables

Es la secuencia ordenada de estados producidos tras cada sentencia atómica ejecutada

La programación concurrente

- siempre mejora los tiempos en sistemas monoprocesador
- mejora el uso de memoria en sistemas monoprocesador si algún proceso se bloquea temporalmente
- mejora los tiempos en sistemas monoprocesador si algún proceso se bloquea temporalmente
- no puede mejorar los tiempos en sistemas monoprocesador

Respecto a un programa concurrente que utiliza un monitor con variables condición (señales):

- Las variables condición deben ser inicializadas antes de ser utilizadas
- Un proceso bloqueado en la cola del monitor es liberado cuando otro proceso ejecuta un signal
- Los procesos que ejecutan un wait sobre una variable condición siempre quedarán bloqueados
- El resultado del programa será el mismo independientemente de la semántica de las señales

Respecto al mecanismo de los cerrojos para solucionar el problema de la EM:

- No cumple la propiedad de progreso
- Utiliza espera bloqueada
- Se basa en el uso de registros hardware específicos para EM
- Se basa en el uso de instrucciones maquina diseñadas especialmente para resolver la exclusión mutua

Dado un algoritmo de exclusión mutua para dos procesos que son bucles infinitos (PE+SC+RS), supongamos que dicho algoritmo cumple todas las propiedades salvo la de progreso; en este caso:

- Un proceso podría quedar indefinidamente en el PE si el otro termina el programa o queda indefinidamente en RS
- Los dos procesos no pueden estar simultáneamente en el PE
- Ambos procesos podrían estar en SC de manera simultánea
- Puede ocurrir que un proceso permanezca en el PE mientras el otro accede a SC varias veces

Dado el siguiente fragmento de código:

```

process Ejecutor;
begin
    receive( tarea, Agregador);
    while tarea != fin do
    begin
        resultado := procesa(tarea) ;
        send (resultado, Agregador);
        receive( tarea, Agregador);
    end
end

```

Podemos identificar que se corresponde con:

- El patrón de un proceso de tipo servidor bajo el paradigma Cliente-Servidor
- El patrón de un proceso de tipo cliente bajo el paradigma Cliente-Servidor
- El patrón de un proceso de tipo maestro bajo el paradigma Maestro-Esclavo
- El patrón de un proceso de tipo esclavo bajo el paradigma Maestro-Esclavo

Dado el siguiente ejemplo de paso de mensajes entre dos procesos:

<pre> process P0 ; var dato : integer; begin dato := 10; enviar(dato, P1); dato := 15; end </pre>	<pre> process P1 ; var dato : integer begin dato := 20; recibir(dato, P0); imprime(dato); end </pre>
---	--

Señalar la opción correcta:

- Si el comportamiento es seguro, P1 imprimirá 15
- ✓ - Si el comportamiento no es seguro, P1 sólo puede imprimir 20 ó 15
- Si el comportamiento no es seguro, P1 puede imprimir 10, 20 ó 15
- Si el comportamiento es seguro, P1 imprimirá 20

En un sistema de tiempo real en el que se dispone de un procesador y hay que ejecutar tres tareas periódicas t1, t2 y t3, con tiempos de ejecución (C) 3, 4 y 6 y períodos (T) 6, 12 y 12, respectivamente.

Llamamos TM a la duración del ciclo principal. Bajo planificación cíclica:

- TM = 24

- ✓ - No sería planificable
- ✓ - $T_M = 12$
- $T_M=72$

Dado el siguiente fragmento de código:

```
process Agregador ;
begin
    for i := 0 to num_procesadores-1 do
        trabajo := escoger_trabajo();
        send( trabajo, Procesador[i] ); { enviar trabajo a procesador }
    while hay trabajo pendiente do begin
        select
            for j:=0 to num_procesadores-1 when receive( resultado, Procesador[j] ) do
                if hay trabajo pendiente then
                    send( escoger_trabajo(), Procesador[j] )
                else
                    send( fin, Procesador[j] );
            end
        procesa(resultados);
    end
end
```

Podemos identificar que se corresponde con

- El patrón de un proceso de tipo servidor bajo el paradigma Cliente-Servidor
- El patrón de un proceso de tipo cliente bajo el paradigma Cliente-Servidor
- ✓ - El patrón de un proceso de tipo maestro bajo el paradigma Maestro-Esclavo
- El patrón de un proceso de tipo esclavo bajo el paradigma Maestro-Esclavo

Al inicio de la ejecución de una sentencia select, suponiendo que hay alternativas con guardas potencialmente ejecutables, pero no hay ninguna alternativa con una guarda ejecutable

- 
- la situación descrita no puede darse nunca por el diseño de la sentencia select
 - la sentencia select acaba sin ejecutar nada
 - el proceso quedará bloqueado a la espera de un envío de uno de los procesos nombrados en las guardas potencialmente ejecutables
 - se selecciona una de las guardas potencialmente ejecutable y se ejecuta inmediatamente, sin esperar

En programación distribuida con paso de mensajes, dos procesos en dos ordenadores distintos se pueden sincronizar mediante el uso de:

~~Paso de mensajes, usando send para envío e i_receive para recepción~~

~~Semáforos en memoria compartida~~

✓ ~~Paso de mensajes, usando s_send para envío y receive para recepción~~

~~Paso de mensajes, usando i_send para envío e i_receive para recepción~~



Ábrete la Cuenta Online de BBVA y
llévate 1 año de Wuolah PRO

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publ en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publi
WUOLAH PRO

Si dos procesos quieren tener una cita mediante el envío y recepción de un único mensaje, debe usar

- Cualquier combinación de modalidades de envío/recepción constituye una cita
- Variables compartidas entre el proceso emisor y el receptor
- Envío síncrono (`s_send`) y recepción insegura (`i_receive`)
- Envío síncrono (`s_send`) y recepción síncrona (`receive`)

~~En un monitor con una semántica de señalar y esperar (SE) y sobre el que se han declarado 3 variables de tipo condición:~~

~~El número total de colas de procesos que gestiona el monitor es 3~~

~~El número total de colas de procesos que gestiona el monitor es 2~~

El número total de colas de procesos que gestiona el monitor es 4

~~El número total de colas de procesos que gestiona el monitor es 5~~

Respecto a la orden de espera selectiva select:

- Si no hay guardas ejecutables ni potencialmente ejecutables, select aborta el programa
- Todas las alternativas deben incluir explícitamente una expresión lógica y una operación receive
- ✓ - Si solamente hay guardas potencialmente ejecutables, select bloquea al proceso que la ejecuta
- Al ejecutar select, únicamente se evalúan las condiciones de alternativas con receive

La sentencia select permite esperar hasta recibir un mensaje de uno de varios posibles emisores, la espera asociada

- es siempre una espera bloqueada
- puede ser una espera ocupada o una espera bloqueada
 - termina cuando se recibe un mensaje de un proceso de máxima prioridad
 - es siempre una espera ocupada

En programación distribuida, la comunicación de datos entre procesos se consigue mediante el uso de:

- Un sistema de envío de mensajes a monitores con esperas bloqueantes
- Un sistema de paso de mensajes entre procesos que se ejecutan en el mismo o en distintos ordenadores

- Un sistema de paso de mensajes que use exclusivamente operaciones de envío y recepción sincronas
- Un sistema de paso de mensajes entre procesos que se ejecutan en distintos ordenadores

Respecto al envío y recepción de un mensaje con send (envío asíncrono seguro) y recepción usando receive

- El proceso emisor no termina send hasta que el receptor que haya escrito los bytes a enviar
- El proceso emisor debe llamar a send antes que el emisor invoque el receive
- (-)** El proceso emisor no termina send hasta que se hayan leído (por el sistema de paso de mensajes) los datos a enviar
- El proceso emisor no termina send antes que el receptor haya llamado a receive

Respecto al envío y recepción de un mensaje con send (envío asíncrono seguro) y receive

- (-)** La operación send esperará al inicio del receive solo cuando el sistema de paso de mensaje no tenga en memoria suficiente para copiar la variable origen
- Si el mensaje no está disponible, la operación de recepción puede bloquearse o no al proceso receptor
- Una vez finalizada la operación de envío, siempre se puede asegurar que el receptor ha recibido el mensaje
- El emisor y el receptor siempre se bloquean esperándose mutuamente

En RPC(Remote Procedure Call)

- El el proceso llamador y el proceso llamado ejecutan código mientras se devuelven los resultados del procedimiento a ejecutar
- Existe un módulo o componente stub en el lado del cliente que se encarga principalmente del envío y recepción de datos a través de la red mediante un producto conocido denominado XDR (external Date Representation)
- Cuando el procedimiento remoto termina, el proceso llamado obtiene el parámetro y continúa ejecutándose el código tras la llamada
- (-)** Existen módulos stubs en los nodos cliente y servidor que se encargan de serializar o hacer marshalling de los datos a intercambiar en un formato conocido denominado XDR (eXternal Data Representation)

Es un envío de un mensaje en el cual usa send o s_send, y el receptor usa receive:

- (-)** Una vez finalizada la operación de envío siempre se puede asegurar que el receptor ha recibido el mensaje
- (-)** Si el receptor no está preparado para recibir el mensaje, el emisor podrá bloquearse o no
- Si el mensaje no está disponible, la operación de recepción puede bloquearse o no al proceso receptor
- El emisor y el receptor siempre se bloquean esperándose mutuamente

Supongamos que en un proceso emisor quiere enviar un mismo mensaje a todos y cada uno de un conjunto de procesos receptores :

- -Es necesario ejecutar múltiples operaciones de envío a un buzón de tipo muchos a muchos
- En algunos sistemas , es posible usar una única operación de envío a un buzón de tipo uno a muchos
- La única forma de conseguirlo es ejecutando múltiples operaciones similares de envío en un bucle , una por cada receptor
- Es imposible implementar esto en sistemas de paso de mensajes

En RMI(Remote Method Invocation)

- Al igual que en RPC(Remote Procedure Call), existen módulos stubs en los nodos cliente y servidor que se encarga de acceder a otros módulos comunicación para el envío y recepción de datos mediante protocolos de red
- Al igual que en RPC (Remote Procedure Call) un objeto puede invocar la ejecución de un método de un objeto remoto bajo el paradigma de iteración síncrona
- Al igual que en RPC (Remote Procedure Call) el objeto remoto debe identificar el nodo o proceso en el que reside el método a ejecutar
- ✓ - Al igual que en RPC(Remote Procedure Call), existen módulos stubs en los nodos cliente y servidor que se encarga de serializar o hacer un marshalling de los datos a intercambiar en un formato conocido denominado XDR

El representante del cliente empaqueta todos los datos de la llamada (nombre del procedimiento y sus parámetros) usando un determinado formato para formar el cuerpo del mensaje a enviar (es usual utilizar el protocolo XDR, eXternal Representation). Este proceso se suele denominar marshalling o serialización.

En un sistema de tiempo real en el que se dispone de un procesador y hay que ejecutar tres tareas periódicas t_1, t_2 y t_3 , con tiempos de ejecución (c_i) 3,4 y 6 , y periodos (T_i) 6 ,5 y 12 , respectivamente ($D_i=T_i$) .Bajo planificación cíclica:

$$-T_s=60$$

$$\begin{aligned} T_m &= \text{lcm}(6,5,12) = 60 \\ T_m &= kT_s \text{ con } k = 5 \Rightarrow T_s = T_m/k = 60/5 = 12 \end{aligned}$$

$$-T_s=24$$

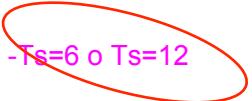
$$-T_s=12$$

-No sería planificable

En un sistema de tiempo real en el que se dispone de un procesador y hay que ejecutar tres tareas periódicas t_1, t_2 y t_3 , con tiempos de ejecución (c_i) 3,4 y 6 , y periodos (T_i) 24 ,12 y 12 , respectivamente ($D_i=T_i$) .Bajo planificación cíclica, T_s puede ser :

$$-T_s=6 \text{ o } T_s=24$$

- $T_s=12$ o $T_s=24$


- $T_s=6$ o $T_s=12$

-No sería planificable

Un sistema de tiempo real:

- Se planifica considerando tiempos promedio de los periodos (T) de todas las tareas a ejecutar dentro de un plazo de tiempo definido
- Se caracteriza por ser no determinista, aunque su ejecución debe garantizarse dentro de un plazo de tiempo definido.
- Se caracteriza por ser determinista y su ejecución debe garantizarse dentro de un plazo de tiempo definido
-  - Se planifica considerando tiempos promedios de cómputo (C) de todas las tareas a ejecutar dentro de un plazo de tiempo definido

Supongamos que un proceso P1 inicia la ejecución de A; send(v,P2); al tiempo que P2 inicia la ejecución de l_receive(v,P1);B; (utilizando recepción insegura). Entonces

- La sentencia A termina de ejecutarse después del inicio de la sentencia B
- La sentencia A no inicia su ejecución antes del fin de B
- La sentencia B no inicia su ejecución antes del fin de A

 - No se puede decir nada del orden relativo de A y B

Dado el siguiente fragmento de código:

```
process Aplicación ;  
begin  
    while true do  
        select  
            for i:= 0 to n-1  
                when condicion[i] receive( petición, Proceso[i] ) do  
                    respuesta := procesar( petición ) ;  
                    s_send( respuesta, Proceso[i] ),  
                end  
            end
```

podemos identificar que se corresponde con:

- El patrón de un proceso de tipo maestro bajo el paradigma de Maestro-Esclavo
- El patrón de un proceso de tipo esclavo bajo el paradigma Maestro-Esclavo
- El patrón de un proceso de tipo cliente bajo el paradigma Cliente-Servidor
-  - El patrón de un proceso de tipo servidor bajo el paradigma Cliente-Servidor

Supongamos un semáforo S inicializando a 0 al cual acceden dos procesos A y B .El proceso A ejecuta un bucle infinito de 5 iteraciones , en cada iteración ejecuta sem_signal(s) , y el proceso B ejecuta un bucle finito de 10 iteraciones , en cada iteración ejecuta sem_wait(s). Entonces

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

- El proceso B puede quedarse bloqueado de vez en cuando , pero siempre saldrá de este bloqueo después de un tiempo finito
- El proceso A puede quedarse bloqueado de vez en cuando , pero siempre saldrá de ese bloqueo después de un tiempo finito
- **Con seguridad el proceso B quedará bloqueado para siempre, sin poder acabar el bucle**
- Es imposible que el proceso B se bloquee en ningún momento , ni siquiera durante un tiempo

~~Respecto a un programa concurrente que utiliza un monitor con variables condición (señales) :~~

- Si se ejecuta un send sobre una variable condición, siempre se liberará algún proceso
- El resultado del programa será el mismo independientemente de la semántica de señales
- **Los procesos que ejecuten un wait sobre una variable condición siempre quedarán bloqueados**
- Un proceso bloqueado en una variable condición es liberado cuando el monitor se queda libre

~~Supongamos un semáforo s inicializado a 0 al cual acceden dos procesos A y B. El proceso A ejecuta un bucle finito de 5 iteraciones, en cada iteración ejecuta sem_signal(s), y el proceso B ejecuta un bucle finito de 10 iteraciones, en cada iteración ejecuta sem_wait(s). Entonces~~

- El proceso B puede quedarse bloqueado de vez en cuando, pero siempre saldrá de ese bloqueo después de un tiempo finito
- El proceso A puede quedarse bloqueado de vez en cuando, pero siempre saldrá de ese bloqueo después de un tiempo finito
- **con seguridad el proceso B quedará bloqueado para siempre , sin poder acabar el bucle**
- Es imposible que el proceso B se bloquee en ningún momento, ni siquiera durante un tiempo.

Test de conceptos SCD T1-T4

(T1) 1. Indicar cuál de los siguientes es la que mejor se adapta al concepto de vivacidad(ausencia de *inanición*) de un proceso que se ejecuta como parte de un programa concurrente:

- Nunca se puede bloquear durante su ejecución

- b. Si el código del proceso contiene instrucciones condicionales, hemos de poder asegurar que dichas instrucciones se ejecutarán frecuentemente si las condiciones de las que depende consiguen un valor de verdad favorable muy a menudo.
- c. **El proceso no puede sufrir un retraso indefinido en la ejecución de sus instrucciones, aunque no podemos dar una indicación precisa de cuándo terminará dicho retraso**
- d. Esta condición se produce cuando se programan bucles de espera activa y no se puede saber cuál de los procesos que llegan al bucle conseguirá terminarlo primero

(T1) 2. Seleccionar la única respuesta correcta. Un programa concurrente que cumpla la propiedad de seguridad para todas sus ejecuciones se considerará correcto si además:

- a. Todos sus procesos también cumplen la propiedad de vivacidad, lo que es equivalente a afirmar que los procesos del programa nunca pueden llegar a una situación de interbloqueo
- b. **Se puede demostrar que sus procesos no sufren inanición en ninguna posible ejecución del programa**
- c. Sus procesos consiguen ejecutar sus instrucciones de forma equitativa
- d. Se puede demostrar que sus procesos no sufren inanición en ninguna posible ejecución del programa y además se ha de cumplir la propiedad de vivacidad

(T1) 3. Seleccionar la única respuesta correcta sobre el concepto de condición de carrera de los programas concurrentes:

- a. Se refiere a que no podemos saber cuál de los procesos del protocolo para resolver el problema de la exclusión mutua conseguirá entrar primero en sección crítica
- b. Cuando se produce no se cumple la propiedad de vivacidad de algunos procesos del programa concurrente
- c. **Múltiples procesos consiguen llegar a cumplir esta condición si el resultado de sus cálculos depende del orden en que se ejecuten**
- d. Esta condición se produce cuando se programan bucles de espera activa y no se puede saber cuál de los procesos que llegan al bucle conseguirá terminarlo primero

(T1) 4. Indicar cuál de las siguientes definiciones se aplica al concepto de programa concurrente correcto (las demás definiciones no consiguen explicar este concepto de una forma totalmente satisfactoria)

- a. Si se puede demostrar que el programa cumple las propiedades de seguridad y vivacidad de los procesos.
- b. Si todos los procesos acceden a recursos compartidos con *justicia* y se obtiene un resultado secuencial correcto.
- c. Si se puede demostrar que todos los procesos del programa cumplen todas las propiedades concurrentes siguientes: *seguridad, vivacidad y equidad*
- d. Si podemos demostrar que todos los procesos terminan sus cálculos y, por tanto, es equivalente a una determinada secuencia de ejecución secuencial

(T1) 5. ¿En qué consiste la ventaja de la programación concurrente si sólo tenemos un procesador en nuestro computador? (justificar la respuesta elegida):

- a. Desacopla la ejecución de los procesos dedicados a entradas/salidas
- b. Se puede crear una hebra de ejecución independiente para realizar los cálculos de cada petición que se reciba de un programa cliente que se ejecuta en otro procesador
- c. El programa tardará menos en ejecutarse porque ahora no se para nunca en las operaciones de entrada/salida
- d. Los compiladores de los lenguajes con procesos concurrentes generan código optimizado

(T1) 6. La verificación utilizando la lógica de programas (pre, post-condiciones e invariantes) no se puede aplicar directamente a la verificación de los programas concurrentes por una de las razones siguientes(justificar la respuesta elegida):

- a. La lógica de programas sólo tiene aplicación a la demostración de corrección de un algoritmo secuencial porque, a diferencia de los programas concurrentes, siempre termina y producen unos resultados
- b. Porque falta una regla de verificación para componer las demostraciones secuenciales de los procesos del programa concurrente, que realizamos utilizando la lógica de programas
- c. Porque aplicando la lógica de programas para verificar las instrucciones de los procesos de un programa concurrente no se pueden demostrar propiedades de vivacidad de dichos procesos
- d. Puede existir interferencia entre los predicados (pre y post-condiciones) con que anotamos las instrucciones que programamos en cada proceso concurrente del programa y la ejecución de instrucciones atómicas de los otros procesos

(T1) 7 En el modelo abstracto de la programación concurrente se incluye la condición de "progreso finito" de los procesos de un programa concurrente. Seleccionar cuál de las siguientes afirmaciones sobre dicha hipótesis es cierta.(justificar la respuesta elegida):

- a. Los procesos de un programa concurrente nunca entrarán en bucles indefinidos de espera activa
- b. Los procesos de un programa concurrente no se pueden suspender o bloquear en ningún caso
- c. **Los procesos de un programa concurrente siempre tienden a ejecutar las instrucciones de su código, pero esto no nos asegura que siempre consigan avanzar en los cálculos que han de realizar**
- d. Es equivalente a afirmar que los procesos de un programa concurrente nunca puedan sufrir inanición

(T2) 1. Seleccionar la única respuesta correcta de las siguientes.

- a. Un programa con procesos concurrentes y programado con monitores no puede implementarse utilizando procesadores multinúcleo pues estos poseen cachés privados
- b. La protección automática de las variables permanentes declaradas dentro de un monitor hace innecesaria la demostración de la propiedad concurrente denominada seguridad de los programas concurrentes
- c. La planificación FIFO obligatoria de las colas de los monitores asegura en todo caso que siempre se satisfaga la propiedad de vivacidad de los procesos de un programa concurrente
- d. **Como consecuencia de haber realizado una llamada a un procedimiento de un monitor, un proceso concurrente se puede suspender y salir del monitor**

(T2) 2. La interpretación correcta del axioma de la operación de sincronización "c.signal()" con señales desplazantes {not empty(c) and L and C} c.signal() {IM and L } es una de las siguientes (Nota: IM= invariante de monitor, L= invariante de las variables locales al procedimiento, C= condición de sincronización para la variable condición implicada).

- a. El proceso concurrente que ocasiona la ejecución de la operación de sincronización no se suspende salvo que la cola "c" esté vacía
- b. El proceso concurrente se suspende en la cola de entrada al monitor cuando ejecuta la operación de sincronización
- c. Cuando se ejecuta la operación de sincronización dentro de un procedimiento del monitor, el estado reentrant del procedimiento hace cierta la condición "C"
- d. **Si la cola no está vacía, el valor de certeza de la condición "C" se transmite a todo el monitor hasta que se desbloquea 1 proceso de la cola**

(T2) 3. La interpretación correcta del axioma de la operación de sincronización c.wait() con señales desplazantes ({ IM and L } c.wait() { C and L }) es una de las siguientes.

- a. **Si no se cumple la condición "C", al volver la llamada a la operación de sincronización, el procedimiento del monitor no se ha programado correctamente**
- b. El proceso que ocasiona la ejecución de la operación se bloquea y deja libre el monitor, cumpliéndose a continuación la condición "C" como precondition de la siguiente instrucción
- c. La condición "C" coincide con el invariante global del monitor
- d. Si no se cumple la condición "C", al volver la llamada a la operación de sincronización, no podemos afirmar que el invariante del monitor se cumpla

(T2) 4. Seleccionar la única respuesta correcta:

- a. Monitores con métodos largos (mucho código) ocasionan que la ejecución de un programa concurrente sea prácticamente secuencial, ya que sólo un proceso concurrente puede entrar al monitor cada vez
- b. **Un proceso de un programa concurrente podría llegar a bloquearse antes de que termine la ejecución del método del monitor al que previamente ha llamado**

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publ en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publ

WUOLAH PRO

- c. Los métodos de los monitores no admiten paso de parámetros por referencia por que se podrían pasar variables globales del programa como argumentos
- d. A diferencia de lo que ocurre con las clases en los lenguajes de programación orientados a objetos, los monitores no se pueden instanciar ya que el invariante ha de ser único

(T2) 5. Seleccionar la única respuesta correcta:

- a. Un programa con procesos concurrentes y programado con monitores no puede implementarse en procesadores multinúcleo- sólo se puede en monoprocesadores con memoria globalmente consistente
- b. **Como consecuencia de la llamada al procedimiento de un monitor, un proceso de un programa concurrente se puede suspender y salir del monitor**
- c. La protección automática de las variables permanentes declaradas dentro de un monitor hace innecesaria la demostración de la propiedad de seguridad en los programas
- d. La planificación FIFO obligatoria de las colas de los monitores asegura en todos los casos que se satisface la propiedad de vivacidad de los procesos de un programa concurrente

(T2) 6. Aunque un monitor garantiza el acceso en exclusión mutua a sus variables permanentes, los métodos de una clase-monitor han de cumplir la propiedad denominada reentrancia (un procedimiento puede ser interrumpido en medio de su ejecución y volver a ser llamado después por otros procesos del programa de manera segura). Seleccionar una de las razones siguientes para que un lenguaje con monitores exija la propiedad de reentrancia:

- a. **Porque en los métodos de cada monitor se pueden programar varias operaciones c.wait()**
- b. Porque durante la ejecución de un programa, los procesos concurrentes van alternándose en la entrada al monitor
- c. Sólo porque un determinado proceso podría entrar varias veces en un mismo monitor y tener aún pendiente la terminación de alguna de sus llamadas anteriores
- d. Porque se pueden anidar las llamadas a los procedimientos de un mismo monitor y un proceso podría tener pendientes de terminación más de 1 llamada

(T2) 7. Sobre el significado de la semántica de las operaciones de variables condición de los monitores, seleccionar la única respuesta correcta:

- a. Las "señales automáticas" se consideran señales desplazantes pero no hay que demostrar la corrección de las operaciones "c.signal()"
- b. Con la semántica denominada "señalar y continuar" (SC) se puede producir "robo de señal" (un tercer proceso entra en el monitor antes que el proceso señalado y cambia el valor de la condición de desbloqueo) pero sólo si todos los procesos se bloquean en la misma cola
- c. **Con semántica de "señales urgentes" puede ocurrir que la salida de un proceso del monitor provoque que un proceso suspendido en una cola del monitor vuelva a ejecutarse**
- d. Con semántica de señales "señalar y salir" (SX) se obliga a que las operaciones c.signal() se programen necesariamente como la última instrucción de los procedimientos del monitor

(T2) 8. Supongamos un algoritmo de exclusión mutua (para dos procesos) que cumple todas las propiedades excepto la de espera limitada, siendo ambos procesos bucles infinitos (PE+SC+RS). Entonces:

- a. Ambos procesos pueden permanecer en PE indefinidamente
- b. Un proceso podría quedar en PE indefinidamente mientras el otro está en RS
- c. **Puede ocurrir que un proceso permanezca en PE mientras el otro accede a SC varias veces**
- d. Todas las anteriores son falsas

(T2) 9. En un programa concurrente con semáforos, en un momento determinado hay dos procesos bloqueados en el semáforo sem y otro proceso hace un sem_signal(sem). El efecto de dicho sem_signal es:

- a. Liberar a uno de los procesos bloqueados en sem
- b. Incrementar el valor del semáforo en una unidad
- c. Liberar a los dos procesos bloqueados en sem
- d. **Liberar a un proceso bloqueado en sem e incrementar el valor del semáforo en una unidad**

(T2) 10. En un monitor ya inicializado y que está siendo utilizado por varios procesos:

- a. La cola del monitor puede estar vacía o tener como máximo un proceso
- b. Si hay un proceso dentro del monitor, todos los demás procesos estarán esperando en la cola del monitor
- c. Si la cola del monitor está vacía, significa que hay un proceso dentro del monitor
- d. **Todas las anteriores son falsas.**

(T2) 11. En un programa concurrente que utiliza el semáforo sem:

- a. **Si hay algún proceso en la cola del semáforo sem, el valor de dicho semáforo es cero en ese momento**
- b. Por la cola de procesos del semáforo sem pasan todos los procesos que han ejecutado un sem_wait(sem)
- c. Un proceso no puede ejecutar dos operaciones sem_wait seguidas sobre el semáforo sem
- d. Todas las anteriores son verdaderas.

(T2) 12. En un monitor con una semántica de señales “señalar y esperar” (SE) y sobre el que se han declarado 2 variables de tipo condición:

- a. El número total de colas de procesos que gestiona el monitor es 4
- b. El número total de colas de procesos que gestiona el monitor es 5
- c. **El número total de colas de procesos que gestiona el monitor es 3**
- d. El número total de colas de procesos que gestiona el monitor es 2

(T2) 13. Supongamos un algoritmo de exclusión mutua para dos procesos 1 y 2, que cumple la propiedad de exclusión mutua y que está diseñado de forma tal que, cuando uno de ellos entre a SC, el otro (aunque esté en RS) tiene garantizado que accederá a SC antes de que el primero vuelva a entrar en la SC:

- a. El algoritmo no cumple la propiedad de espera limitada
- b. El algoritmo no cumple la propiedad de progreso en la ejecución
- c. **El algoritmo puede producir interbloqueo**
- d. El algoritmo es correcto

(T2) 14. En un programa concurrente que utiliza el semáforo sem, que está inicializado a cero:

- a. La primera operación sem_wait sobre el semáforo sem bloqueará siempre al proceso que la ejecuta
- b. La primera operación sem_signal sobre el semáforo sem no tiene ningún efecto
- c. No se puede hacer un sem_wait(sem) hasta que no se haga un sem_signal(sem)
- d. Todas las anteriores son falsas

(T2) 15. Respecto al concepto de monitor:

- a. Las variables permanentes pueden ser accedidas fuera del monitor, siempre que no se modifique su valor
- b. Si un procedimiento está ejecutando un procedimiento del monitor (A), ningún otro proceso podrá ejecutar el procedimiento A pero sí otro procedimiento de dicho monitor
- c. Si un procedimiento está ejecutando un procedimiento del monitor (A), ningún otro proceso podrá ejecutar el procedimiento A ni ningún otro procedimiento de dicho monitor
- d. La a) y la b) son correctas

(T2) 16. Respecto a los cerrojos, señalar la opción correcta:

- a. Los cerrojos son un mecanismo que no usa espera ocupada
- b. Los cerrojos utilizan instrucciones máquina atómicas
- c. La instrucción “LeerAsignar(a)” utiliza una variable booleana
- d. La b) y la c) son correctas

(T2) 17. Respecto a un programa concurrente que utiliza un monitor:

- a. Todas las hebras del programa deben llamar a algún procedimiento del monitor
- b. Lo único que las hebras del programa pueden hacer sobre el monitor es invocar a los procedimientos exportados de dicho monitor
- c. El monitor es un proceso que se ejecuta concurrentemente con el resto de hebras del programa
- d. Todas las anteriores son falsas

Temas 3 y 4

(T3 y T4) 1. Seleccionar la alternativa verdadera:

- cuándo?
- a. En ningún caso el proceso receptor se suspenderá al ejecutar la orden receive(...) en el paso de mensajes asíncrono con búfer
 - b. Las condiciones de la instrucción de espera selectiva (select) han de ser excluyentes entre las distintas alternativas de esta orden
 - c. Una orden select podría suspender su ejecución incluso si todas las condiciones de sus alternativas se hubieran evaluado como ciertas
 - d. El paso de mensajes síncrono no puede implementarse con un búfer

fue por
descarte, la
única sino que
me cuadra es
la b

a. (T3 y T4) 2. Seleccionar la alternativa verdadera:

- a. El algoritmo RMS nos dice que la planificabilidad (las tareas consiguen ejecutar sus unidades antes de expire su plazo máximo de respuesta) de un conjunto de N tareas periódicas es imposible si la utilización conjunta del procesador supera el límite:

$$U=N*(2^{(1/N)}-1)$$

- b. Si se utiliza asignación estática de prioridades a las tareas, entonces podemos afirmar que dicho conjunto de tareas es planificable independientemente de la fase de cada tarea

c. **El plazo de respuesta máximo (D) para cada tarea no depende del instante de su próxima activación**

- d. El algoritmo de planificación denominado RMS nunca puede proporcionarnos un esquema de planificación de tareas con aprovechamiento del 100% del tiempo del procesador

(T3 y T4) 3. Seleccionar las afirmaciones correctas respecto de la ejecución de las alternativas de la espera selectiva `select` que poseen algunos lenguajes de programación distribuida:

- a. Si hay guardas ejecutables en las alternativas, se selecciona no determinísticamente una entre las que poseen una orden `send` ya iniciada

b. **Si no hay guardas ejecutables pero sí las hay potencialmente ejecutables, la instrucción de espera selectiva se suspende hasta que un proceso nombrado inicie una operación `send`**

- c. Si en las alternativas no se ha programado ninguna sentencia de entrada (`receive`), se selecciona no determinísticamente una cualquiera de éstas para su ejecución y la espera selectiva termina

- d. La espera selectiva nunca puede levantar una excepción en el programa donde se programe

(T3 y T4) 4. Seleccionar la afirmación correcta respecto de la espera selectiva con guardas indexadas

a. **Las condiciones de las alternativas de la espera selectiva no pueden depender de argumentos de entrada (arg) de las sentencias de entrada (`receive (var arg)`) que se programen en éstas**

- b. El índice que se programa para replicar una alternativa no puede depender de valores límite (inicial, final) no conocidos en tiempo de compilación del programa

- c. En la instrucción de espera selectiva no se pueden combinar alternativas indexadas con otras alternativas normales no indexadas

- d. Un conjunto de N procesos emisores cada uno envía caracteres al resto de los procesos de dicho conjunto, es decir, cada proceso recibe (N-1) mensajes de los demás, entonces es imposible programar los procesos individuales con instrucciones de espera selectiva porque dicha orden se ejecuta 1 vez y termina

(T3 y T4) 5. Seleccionar la afirmación correcta

- a. La instrucción de espera selectiva no es necesaria en los lenguajes de programación porque todo se puede programar con operaciones de paso de mensajes no bloqueantes

- b. `MPI_Probe` o comprobación bloqueante de mensaje es redundante con `MPI_Wait`

c. **`MPI_Send` (con soporte hardware) podría volver sin esperar la ejecución de la operación de recepción concordante**

- d. `MPI_Recv` podría volver sin esperar la ejecución de la operación de envío concordante

(T3 y T4) 6. Seleccionar la única respuesta correcta respecto de la orden `MPI_Receive`:

- a. El proceso que programe dicha orden siempre se bloquea independientemente del estado del búfer (vacío o con datos esperando)

- b. Sólo si suponemos un mecanismo de comunicación síncrono sin búfer ("citas") un proceso que programe dicha orden siempre se bloqueará

- c. Si la ejecución de la orden anterior no bloquea al proceso, entonces no podemos asegurar que la operación de transmisión de los datos sea segura

cómo???



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

ventajas

PRO



Di adiós a la publi en los apuntes y en la web



Acumula tickets para los sorteos



Descarga carpetas completas

estudia sin publi

WUOLAH PRO

- d. El proceso que programe dicha orden se bloqueará sólo si el búfer es vacío, es decir, no hay mensajes pendientes de ser recibidos

(T3 y T4) 7. Seleccionar la única respuesta correcta en el caso de que un proceso programe la operación insegura MPI_Irecv(...):

- a. La reducción del tiempo de recepción del mensaje en el receptor es independiente de que el sistema de ejecución cuente con hardware especializado o no

- b. Los datos ya transmitidos siempre se mantienen el búfer del sistema hasta que el proceso receptor pueda descargarlos a su espacio de memoria y esto es independiente de que el proceso que ejecute la orden vuelva inmediatamente (sistema con hardware especializado)

- c. Con esta operación no se iniciará la transmisión de datos entre el proceso emisor y el receptor inmediatamente

- d. Con esta operación siempre se anulará el tiempo de espera en el proceso receptor

(T3 y T4) 8. La elección y ejecución inmediata de 1 alternativa de la orden de espera selectiva (select) sólo se producirá si se cumple una de las condiciones siguientes (indicar cuál):

- a. Sólo depende de que exista alguna orden potencialmente ejecutable en ese momento

- b. Sólo depende de que alguna condición de las órdenes con guarda sea cierta

- c. Existe, al menos, una orden potencialmente ejecutable y además se nombra a un proceso del programa que ya ha iniciado su envío

- d. Sólo de que existe algún proceso del programa que haya iniciado su envío

(T3 y T4) 9. Indicar cuál de las siguientes afirmaciones puede considerarse correcta respecto de la orden de espera selectiva:

- a. Una vez que se ha seleccionado una orden potencialmente ejecutable, el proceso que la contiene se ejecuta secuencialmente hasta que termina el bloque componente (do...end)

- b. Una orden de espera selectiva no termina hasta que no ejecute cada una de sus alternativas, lo cual ocurre no determinísticamente

- c. Una orden de espera selectiva puede entremezclar las secuencias de ejecución de las instrucciones de los bloques de sentencias componentes de cada una de sus alternativas

- d. La orden de espera selectiva puede programar un array de guardas indexadas, pero cada una ha de incluir una operación de entrada (receive(...))

(T3 y T4) 10. Indicar cuál de las siguientes afirmaciones es correcta con respecto a la denominada deriva acumulativa que experimentan los programas de tiempo real:

- a. La deriva acumulativa se produce en una tarea porque no es posible programar un retraso exacto hasta un instante en que produzca la siguiente activación, es decir, todos los retrasos que se programan indican un intervalo de suspensión relativo al instante en que se ejecutan

- b. Se puede eliminar la deriva de los retrasos relativos programados en cada tarea pero nunca se podrán eliminar completamente los retrasos causados por el sistema operativo

- c. La orden sleep_until(...) permite eliminar completamente la deriva acumulativa

- d. Siempre es posible programar una tarea periódica que se active transcurrido un periodo de tiempo exacto desde su última activación en todas sus ejecuciones

(T3 y T4) 11. Respecto del esquema de planificación dinámica de tareas EDF, indicar cuál de las siguientes afirmaciones es la correcta

- a. La aplicación de este método es incompatible con aplicar el RMS

- b. Un conjunto de tareas cuya utilización del procesador es del 100% y que resulta ser planificable con el algoritmo EDF podría ser también planificable con RMS

- c. EDF es el único esquema de planificación dinámico

- d. Siempre podremos planificar con EDF cualquier conjunto de tareas cuya utilización del procesador no sea mayor del 100%, independiente de valor de su plazo de tiempo límite