Cerrar

Comenzado el	lunes, 1 de febrero de 2021, 16:30
Estado	Finalizado
Finalizado en	lunes, 1 de febrero de 2021, 16:59
Tiempo empleado	29 minutos 50 segundos
Calificación	**5,25** de 10,00 (**53**%)

Pregunta **1**
Correcta
Puntúa 0,50 sobre 0,50

Una hebra tiene acceso

Seleccione una:

- a las variables declaradas en el resto de hilos de la aplicación.
- a cualquier variable pública declarada como estática en la aplicación. ✔
- a todas las variables que se encuentran en el heap de la aplicación.
- A las variables públicas declaradas en el resto de procesos en ejecución.

La respuesta correcta es: a cualquier variable pública declarada como estática en la aplicación.

Pregunta **2**
Incorrecta
Puntúa -0,15 sobre 0,50

El uso de ejecutores de pool de threads en ambientes de programación paralela basados en multiprocesamiento simétrico tiene sentido, en términos de reducción de latencias de creación de hebras

Seleccione una:

- siempre. ✖
- en problemas donde la carga de tareas paralelas es predecible, pequeña y constante.
- nunca.
- En problemas donde la carga de tareas paralelas presenta variabilidad en el tiempo, inclusive con picos de alta demanda de tareas, sin un grado de predectibilidad claramente establecido.

La respuesta correcta es: En problemas donde la carga de tareas paralelas presenta variabilidad en el tiempo, inclusive con picos de alta demanda de tareas, sin un grado de predectibilidad claramente establecido.

Pregunta **3**
Correcta
Puntúa 0,50 sobre 0,50

El empleo del mecanismo de comunicación entre procesos basado en el modelo de paso de mensajes es la opción más adecuada cuando

Seleccione una:

- la comunicación y/o sincronización entre los procesos se realiza en un sistema que emplea un esquema de memoria compartida.
- la comunicación y/o sincronización entre procesos se realiza en un sistema encuadrado dentro de la categoría SIMD de la taxonomía de Flynn.
- la comunicación y/o sincronización entre los procesos se realiza en un sistema que emplea un esquema de memoria distribuida. ✔
- se desea comunicar, pero no sincronizar, procesos distribuidos en un sistema encuadrado dentro de la categoría SIMD de la taxonomía de Flynn.

La respuesta correcta es: la comunicación y/o sincronización entre los procesos se realiza en un sistema que emplea un esquema de memoria distribuida.

Correcta

Puntúa 0,50 sobre 0,50

Seleccione una:

- Los procesos no están en la cola de planificación de corto plazo del sistema operativo, sino en una cola de procesos bloqueados y no intentan acceder a la CPU.
- Los procesos quedan completamente bloqueados a la espera de que otro proceso los desbloquee, lo cuál no podrá ocurrir.
- Los procesos siguen dentro de la planificación del sistema operativo y consumen ciclos de CPU. ✔
- Los procesos han intentado modificar una variable simultáneamente, por lo que el sistema queda a la espera de una acción que indique el orden de modificación.

La respuesta correcta es: Los procesos siguen dentro de la planificación del sistema operativo y consumen ciclos de CPU.

Pregunta **5**

Incorrecta

Puntúa -0,15 sobre 0,50

En un sistema de tiempo real

Seleccione una:

- el tiempo de cómputo requerido por una tarea no puede superar nunca el plazo de respuesta máximo. ✖
- el tiempo de cómputo requerido por una tarea periódica será siempre igual o inferior al periodo de activación de la tarea.
- las tareas se ejecutan de forma secuencial y no es necesario controlar el acceso en exclusión mutua a los recursos compartidos.
- las aplicaciones se ejecutarán habitualmente más rápido que en otro sistema con las mismas prestaciones pero que no imponga una ligadura temporal sobre los procesos.

La respuesta correcta es: el tiempo de cómputo requerido por una tarea periódica será siempre igual o inferior al periodo de activación de la tarea.

Sin contestar

Puntúa como 0,50

```
#include <iostream>
#include <thread>
#include <mutex>
#include <vector>

struct C {
        std::mutex lock;
        int xx = 0;

        void x() {std::lock_guard<std::mutex> guard(lock);
                  xx++;}
        void y() {std::lock_guard<std::mutex> guard(lock);
                  xx--;}
        void z() {lock.lock();
                  xx--;}
};

int main() {
        C cont;
        std::vector<std::thread> threads;
        for(int i=0; i<10; ++i) {
          threads.push_back(std::thread([&cont](){
            for(int j=0; j<100; ++j)
              cont.x();}));
        }
        for(int j=0; j<10; j++)threads[j].join();
          cont.y();
          cont.z();
        std::cout << cont.xx-10 << std::endl;
        cont.z();
        std::cout << cont.xx-10 << std::endl;
        return 0;
}
```

Seleccione una:

- El programa imprime 999, 998 y termina normalmente.
- El programa imprime 999, 989 y termina normalmente.
- El programa imprime 988, 978 y queda bloqueado.
- El programa imprime 988 y queda bloqueado

La respuesta correcta es: El programa imprime 988 y queda bloqueado

Correcta

Puntúa 0,50 sobre 0,50

declarado en la superclase como *synchronized*. La subclase sobreescribe el método. ¿Cuál de las afirmaciones es correcta?

Seleccione una:

- El mecanismo de herencia en el lenguaje Java no se aplica sobre el modificador *synchronized*, de tal forma que será necesario volver a escribir *synchronized* en el método sobreescrito para mantener este tipo de sincronización. ✔
- No será necesario controlar este método en exclusión mutua, ya que al ser un método con la cláusula *synchronized* en la súperclase, este comportamiento también será heredado en las subclases.
- El mecanismo de herencia en el lenguaje Java no permite heredar este tipo de métodos, por lo que ocurrirá un error de compilación.
- Ninguna de las afirmaciones es correcta.

La respuesta correcta es: El mecanismo de herencia en el lenguaje Java no se aplica sobre el modificador *synchronized*, de tal forma que será necesario volver a escribir *synchronized* en el método sobreescrito para mantener este tipo de sincronización.

Pregunta **8**

Sin contestar

Puntúa como 0,50

Considere el siguiente programa escrito en Java. ¿Cuál es su comportamiento en tiempo de ejecución?

```
public class V extends Thread {
 Condition[] c;
 int i;
 ReentrantLock l;

 public V(int i){
   this.i = i;
   l = new ReentrantLock();
   c = new Condition[this.i+1];
   for(int k=0; k<c.length; k++) c[k] = l.newCondition();
 }

 public void run (){
   l.lock();
   try {
     c[this.i].await();
     System.out.println("Soy la hebra "+this.i);
   } catch (InterruptedException e){} finally {l.unlock();}
  }

 public void aux() throws InterruptedException {
   l.lock();
   try {c[this.i].signal();
   } finally{l.unlock();}
  }

public static void main(String[] args) throws Exception{
    V[] t = new V[4];
    int s = 2;
    for(int i=0; i<t.length; i++) t[i]=new V(i);
    for(int i=0; i<t.length; i++) t[i].start();
    Thread.currentThread().sleep(1000);
    t[s].aux();
    for(int i=0; i<t.length; i++) t[i].join();
    System.out.println("Fin del programa...");
 }
}
```

Seleccione una:

- El programa efectúa cuatro impresiones de la forma "Soy la hebra n", donde n es un número entre 1 y 4, a continuación imprime "Fin del programa..." y termina normalmente.
- El programa efectúa varias impresiones de la forma "Soy la hebra n", donde n es un número entre 0 y 3, y queda bloqueado.
- El programa efectúa una impresión de la forma "Soy la hebra 2", y termina normalmente imprimiendo "Fin del programa...".
- Ninguna de las anteriores es correcta.

La respuesta correcta es: Ninguna de las anteriores es correcta.

Correcta

Puntúa 0,50 sobre 0,50

```
import java.util.Scanner;
public class cumbresTenebrosas extends Thread{
  public static int v = 0;
  public static int n = 1000;

  public  cumbresTenebrosas(){this.start();}
  public void Heatcliff(){
    if(n>0){
      n--;
      synchronized(this){
        v--;
        this.Heatcliff();
      }
    }
  }
  public void run(){
    this.Heatcliff();
  }
  public static void main(String[]args){
    cumbresTenebrosas q = new cumbresTenebrosas();
    try{q.join();}catch(InterruptedException e){}
    System.out.println(v);
  }
}
```

Seleccione una:

- Ninguno. El programa se bloquea.
- 1000
- 0
- -1000 ✔

La respuesta correcta es: -1000

Pregunta **10**

Incorrecta

Puntúa -0,15 sobre 0,50

Una aplicación concurrente

Seleccione una:

- puede comunicarse con otros procesos del sistema a través de variables almacenadas en la zona de memoria compartida (variables static).
- siempre consta de al menos de dos tareas en ejecución.
- será siempre más eficiente si emplea un *pool* para crear los hilos.
- Ninguna de las otras respuestas es correcta. ✖

La respuesta correcta es: siempre consta de al menos de dos tareas en ejecución.

Correcta

Puntúa 0,50 sobre 0,50

de la aplicación, usted decide

Seleccione una:

- desarrollar una aplicación multiproceso que emplee el mecanismo de paso de mensajes para sincronizarlos y establecer una comunicación entre ellos. ✔
- desarrollar una aplicación multiproceso que emplee exclusivamente el modelo de memoria común para hebras de Python, a fín de sincronizarlos y establecer una comunicación entre ellos.
- desarrollar una aplicación multihilo que emplee variables compartidas para sincronizarlos y establecer una comunicación entre ellos.
- desarrollar una aplicación que emplee un *pool* para gestionar los hilos de la aplicación y usar variables compartidas para sincronizar y establecer una comcomunicación entre las tareas enviadas al *pool*.

La respuesta correcta es: desarrollar una aplicación multiproceso que emplee el mecanismo de paso de mensajes para sincronizarlos y establecer una comunicación entre ellos.

Pregunta **12**

Incorrecta

Puntúa -0,15 sobre 0,50

Se ha desarrollado con Java una versión paralela beta de un software destinado a simular interacciones intermoleculares entre moléculas de estructuras cristalinas tridimensionales regulares sobre una máquina multicore clasificada MIMD en la taxonomía de Flynn con acoplamiento fuerte y memoria común. Se plantea la explotación científica de la beta en un cluster de procesadores de alto rendimiento, aplicando la misma sobre estructuras cristalinas complejas, que exigen distribuir el cálculo entre los nodos del cluster y una alta tasa de comunicación internodal. En estas condiciones:

Seleccione una:

- La beta es directamente ejecutable sobre el cluster.
- Es necesario reescribir la beta utilizando MPJ-Express para gestionar la alta tasa de transferencia de información entre tareas que sabemos que existe.
- Es necesario reescribir la beta utilizando control de acceso a la información utilizando memoria transaccional software con Java/Clojure.
- Ninguna de las anteriores es correcta. ✖

La respuesta correcta es: Es necesario reescribir la beta utilizando MPJ-Express para gestionar la alta tasa de transferencia de información entre tareas que sabemos que existe.

Sin contestar

Puntúa como 0,50

binario `mpjrun.bat -np 2 xYZ` ¿Cuál es la salida impresa que produce?

```
import mpi.*;

public class xYZ{
public static void main(String args[]) throws Exception {

 MPI.Init(args);
 int rank = MPI.COMM_WORLD.Rank();
 int size = MPI.COMM_WORLD.Size();
 int emisor = 0; int receptor = 1;
 int tag = 100; int unitSize = 1;

 if(rank==emisor){
     int[] v1=new int[3];
     v1[0]=2;
     v1[1]=4;
     v1[2]=6;
     int bufer1[] = new int[3];
     for(int i=0; i<bufer1.length; i++)
         MPI.COMM_WORLD.Send(v1, i, unitSize, MPI.INT, receptor, tag+i);
     MPI.COMM_WORLD.Recv(v1, 0, unitSize, MPI.INT, receptor, 10);
     System.out.println(v1[0]);
 } else{
          int revbufer[] = new int[3];
          for(int i=0; i<revbufer.length; i++)
          MPI.COMM_WORLD.Recv(revbufer, i, unitSize, MPI.INT, emisor, tag+i);
          int []aux= new int[1]; aux[0]=1;
          for(int i=0; i<revbufer.length; i++)
            aux[0]=aux[0]*(revbufer[i]*2);
          MPI.COMM_WORLD.Send(aux, 0, unitSize, MPI.INT, emisor, 10);
   }
 MPI.Finalize();
 }
}
```

Seleccione una:

- 48
- 384
- 128
- 732

La respuesta correcta es: 384

Correcta

Puntúa 0,50 sobre 0,50

ello, se escribe una clase C de Java que encapsula los recursos necesarios (array de tenedores), con el objetivo de poder instanciar a partir de C tantos monitores como se desee, para versiones del problema donde el número de filósofos es variable. Para ello, su equipo de programadores

Seleccione una:

- debe escribir una clase con métodos que utilicen un cerrojo de clase ReentrantLock común a todos ellos, y múltiples condiciones Condition, e incorporar la sincronización estándar específica del problema. ✔
- debe escribir una clase con métodos que utilicen un cerrojo de clase ReentrantLock propio y específico de cada método, y múltiples condiciones Condition, e incorporar la sincronización estándar específica del problema.
- debe escribir una clase cuyos métodos estén cualificados como "synchronized", definir condiciones "Condition" , e incorporar la sincronización estándar necesaria para resolver el problema.
- Todas las respuestas son correctas.

La respuesta correcta es: debe escribir una clase con métodos que utilicen un cerrojo de clase ReentrantLock común a todos ellos, y múltiples condiciones Condition, e incorporar la sincronización estándar específica del problema.

Pregunta **15**

Incorrecta

Puntúa -0,15 sobre 0,50

Considere desde un punto de vista teórico la primitiva de control de la concurrencia conocida como monitor, y suponga que para un determinado problema, las necesidades de sincronización del mismo requieren un sistema de colas de condición compuesto por n unidades. El monitor va a dar soporte a la política de señalización mediante el modelo de señales SU. En estas condiciones, el número total de colas de este monitor es:

Seleccione una:

- igual a n+1.
- igual n+2.
- igual a n+3.
- igual a 2: un wait-set, y una cola de condición que simula al sistema de n colas. ✖

La respuesta correcta es: igual n+2.

Pregunta **16**

Correcta

Puntúa 0,50 sobre 0,50

Considere una clase escrita en C++ en la cuál una parte de sus métodos están sincronizados utilizando un cerrojo de clase mutex, mientras que los métodos restantes están sincronizados utilizando un segundo cerrojo de clase recursive_mutex. Todos los métodos tienen acceso a todos los atributos de la clase y existen métodos de índole recursiva cuya exclusión mutua es controlada bien con mutex estándar, bien mediante mutex recursivo. Bajo las condiciones descritas se crea un objeto de la clase, y se da acceso al mismo a múltiples hebras concurrentes. Podemos entonces asegurar que:

Seleccione una:

- el acceso al objeto es seguro frente a exclusión mutua y bloqueos.
- el acceso al objeto es seguro frente a exclusión mutua pero no frente a bloqueos.
- el acceso al objeto no es seguro ni frente a exclusión mutua ni frente a bloqueos. ✔
- el acceso al objeto es seguro frente a bloqueos pero no frente exclusión mutua.

La respuesta correcta es: el acceso al objeto no es seguro ni frente a exclusión mutua ni frente a bloqueos.

Pregunta **17**

Correcta

Puntúa 0,50 sobre 0,50

Para que un protocolo de exclusión muta sea considerado como correcto, debe

Seleccione una:

- únicamente estar libre de interbloqueo y evitar la inanición de procesos.
- únicamente garantizar el acceso atómico al recurso que protege.
- únicamente evitar condiciones de carrera entre procesos.
- Ninguna de las otras respuestas es correcta. ✔

La respuesta correcta es: Ninguna de las otras respuestas es correcta.

Correcta

Puntúa 0,50 sobre 0,50

Seleccione una:

- debe de ser siempre ejecutado como última instrucción de los métodos públicos de cualquier monitor.
- debe de ser siempre ejecutado como última instrucción de todos los métodos de cualquier monitor.
- no garantiza que todas las hebras del *wait-set* asociado al objeto sobre el que se aplica reciben la señal.
- Ninguna de las otras respuestas es correcta. ✔

La respuesta correcta es: Ninguna de las otras respuestas es correcta.

Pregunta **19**

Correcta

Puntúa 0,50 sobre 0,50

Suponga que está implementando el conocido problema del productor-consumidor en el lenguaje C++. ¿Por qué razón implementaría el proceso consumidor en una λ-función (función lambda)?

Seleccione una:

- Porque todos los recursos que utilice dentro de este tipo de funciones se considerarán atómicos.
- Porque es necesario garantizar que otras hebras no acceden a los recursos compartidos.
- Porque este tipo de funciones son las encargadas arrancar el objeto de la clase thread de forma automática.
- Ninguna de las otras respuestas es correcta. ✔

La respuesta correcta es: Ninguna de las otras respuestas es correcta.

Pregunta **20**

Correcta

Puntúa 0,50 sobre 0,50

Considere un escenario de concurrencia con Java donde se emplean estructuras de datos complejas con una alta carga de hebras concurrentes, del cuál se sabe, dada la natualeza del problema, que el número de accesos concurrentes para lectura de las hebras a las estructuras de datos es mucho mayor que el número de accesos para escritura. En estas condiciones, y para asegurar una solución al escenario que sea segura y optimice la vivacidad usted considera que la mejor solución es:

Seleccione una:

- proteger los accesos a las estructuras de datos mediante semáforos.
- proteger los accesos a las estructuras de datos mediante monitores
- proteger los accesos a las estructuras de datos mediante cerrojos de clase ReentrantLock.
- proteger los accesos a las estructuras de datos mediante transacciones Java/Clojure. ✔

La respuesta correcta es: proteger los accesos a las estructuras de datos mediante transacciones Java/Clojure.

Ir a...

Cuestionario (Test) de Examen de Teoría-Febrero 2021-LLAMAMIENTO ESPECIAL ►

¿NECESITAS AYUDA?

CAU Campus Virtual

campus.virtual@uca.es

Horario de 9:00 a 14:00 L-V
956 01 67 55

Canal YouTube

/