01:04:16

Kiểm tra giữa kỳ (CSE423 - Lập trình Phân tán) - Đề 3

GĐ 1, HK 1, 2022-2023

Xin chào, Dinh Quang. Khi bạn gửi biểu mẫu này, chủ sở hữu sẽ thấy tên và địa chỉ email của bạn.

* Bắt buộc

1

Họ và tên *

Đinh Quang Hiếu

2

Mã sinh viên *

1951060699

3

Lớp *

01:04:16

4

Thuật toán của Peterson cho bài toán loại trừ lẫn nhau trong hệ thống đồng thời thỏa mãn những thuộc tính nào dưới đây? * (1 Điểm)

- [x] **Không chết đói** (starvation-freedom): nếu một luồng đang cố gắng đi vào CS, thì luồng đó cuối cùng phải đi vào CS thành công
- [x] **Tiến triển** (progress): nếu một hoặc nhiều luồng đang cố gắng để đi vào CS và không có luồng nào bên trong CS, thì ít nhất một trong các luồng sẽ thành công trong việc đi vào CS
- [x] **Loại trừ lẫn nhau** (mutual exclusion): hai luồng bất kỳ không thể ở trong khu vực quan trọng (CS) tại cùng một thời điểm
- [ ] Có thể hoạt động được với số lượng luồng bất kỳ lớn hơn 1

5

Trong bài toán Bữa tối của Triết gia, với n > 1 luồng (Triết gia) cùng thực thi, chúng ta cần đảm bảo các điều kiện đồng bộ nào? * (1 Điểm)

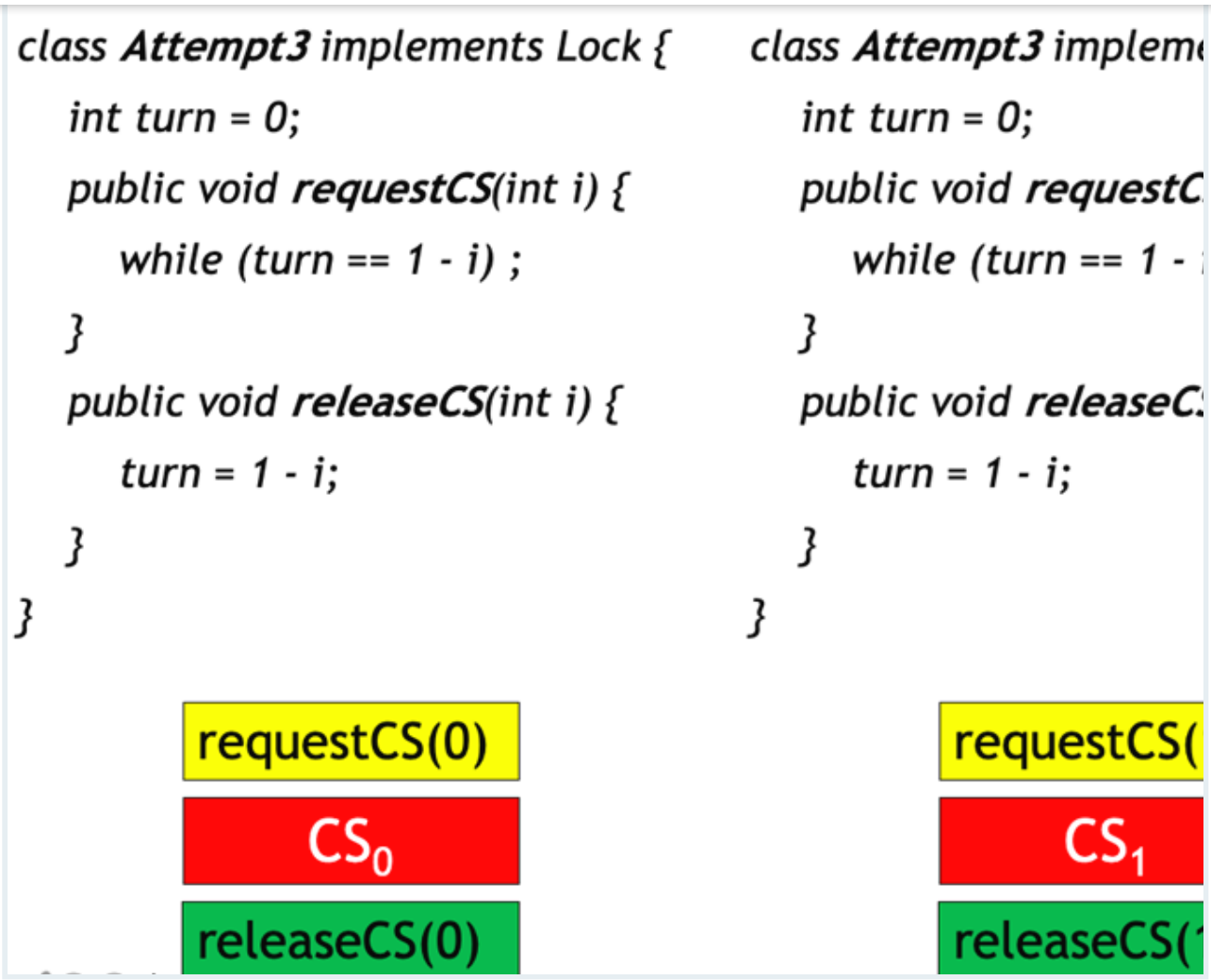
- [x] Để có thể đi vào khu vực quan trọng (i.e., thực hiện việc Ăn mỳ), mỗi luồng (Triết gia) cần phải lấy được 2 tài nguyên bên cạnh (i.e., hai chiếc nĩa ở bên trái và bên phải)
- [ ] Hai luồng (Triết gia) cạnh nhau có thể cùng đi vào khu vực quan trọng (i.e., cùng thực hiện việc Ăn mỳ)

6

Xét thuật toán trong Hình dưới để giải quyết bài toán loại trừ lẫn nhau trong một hệ thống đồng thời có 2 luồng cùng hoạt động.

Những kịch bản nào sau đây là đúng cho thứ tự đi vào khu vực quan trọng của hai luồng T0, T1 nếu hệ thống sử dụng thuật toán này? * (1 Điểm)

01:04:16



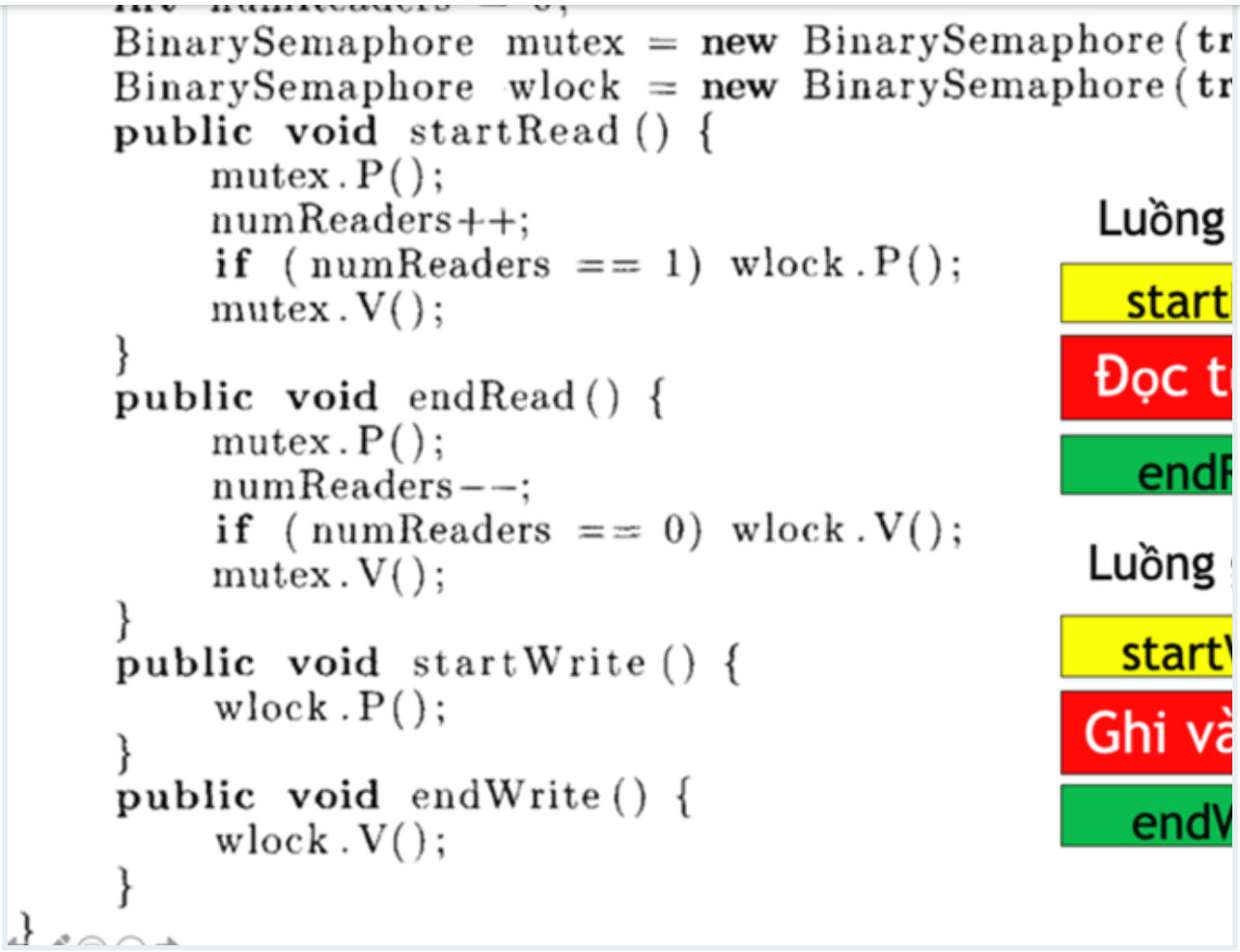
- [x] T0, T1, T0, T1, T0, T1
- [ ] T1, T1, T1, T0, T0, T0
- [x] T1, T0, T1, T0, T1, T0
- [x] T0, T1, T1, T0, T0, T1

7

Xét cài đặt của giải pháp cho bài toán Người đọc - Người ghi, sử dụng cấu trúc đồng bộ hoá Semaphore, như Hình dưới.

Mục đích của dòng lệnh **if (numberReaders == 1) wlock.P();** trong phương thức **startRead()** là gì? * (1 Điểm)

01:04:16



- [ ] Đánh thức một luồng ghi bất kỳ đang bị khoá dậy để thực hiện tiếp công việc của nó
- [ ] Đánh thức một luồng đọc bất kỳ đang bị khoá dậy để thực hiện tiếp công việc của nó
- [ ] Nếu đây là luồng đọc đầu tiên thành công đi vào khu vực quan trọng thì sẽ chuyển semaphore nhị phân wlock sang trạng thái không sẵn sàng để chặn các luồng đọc phía sau đi vào khu vực quan trọng
- [x] Nếu đây là luồng đọc đầu tiên thành công đi vào khu vực quan trọng thì sẽ chuyển semaphore nhị phân wlock sang trạng thái không sẵn sàng để chặn các luồng ghi phía sau đi vào khu vực quan trọng

8

Chương trình đồng thời khác với chương trình tuần tự ở những điểm nào sau đây?

* (1 Điểm)

01:04:16

- [x] Khi chạy một chương trình đồng thời, có thể xảy ra nhiều kịch bản khác nhau, dẫn đến nhiều kết quả khác nhau
- [ ] Hai kiểu chương trình này không có sự khác biệt nào cả
- [x] Trong chương trình đồng thời, tại một thời điểm có thể thực hiện nhiều tính toán, trong khi đó với chương trình tuần tự, tại một thời điểm chỉ có nhiều nhất 1 tính toán được thực hiện

9

Các cấu trúc đồng bộ hoá, như Semaphore, Monitor, có thể được sử dụng để giải quyết những yêu cầu nào sau đây? * (1 Điểm)

- [x] Yêu cầu đồng bộ
- [x] Yêu cầu loại trừ lẫn nhau

10

Semaphore nhị phân gồm có những thành phần nào? * (1 Điểm)

- [x] Thao tác P() được thực thi nguyên tử: dùng để thêm luồng gọi vào hàng đợi nếu semaphore không ở trạng thái sẵn sàng
- [x] Một biến value kiểu boolean
- [x] Một hàng đợi các luồng bị khóa – được khởi tạo là rỗng
- [ ] Một biến value kiểu int
- [x] Thao tác V() được thực thi nguyên tử: dùng để đánh thức một luồng bất kỳ trong hàng đợi

01:04:16

Semaphore đếm khác Semaphore nhị phân như thế nào? * (1 Điểm)

- [ ] Không có thao tác V() trong Semaphore đếm
- [ ] Không có thao tác P() trong Semaphore đếm
- [ ] Không có hàng đợi các luồng bị khóa trong Semaphore đếm
- [ ] Semaphore đếm cho phép nhiều luồng cùng ở trong khu vực quan trọng (CS) trong một thời điểm, trong khi Semaphore nhị phân chỉ cho phép nhiều nhất 1 luồng ở trong CS
- [x] Biến value trong Semaphore đếm có kiểu int, không phải kiểu boolean như trong Semaphore nhị phân

12

Cho đoạn mã giả sau, cài đặt giải pháp cho bài toán Người đọc - Người ghi, trong đó có n > 2 luồng đọc và m > 2 luồng ghi, cùng tương tác với cơ sở dữ liệu chia sẻ.

Nếu các khối lệnh cho việc **<< GHI DỮ LIỆU VÀO DB>>** và **<< ĐỌC DỮ LIỆU TỪ DB>>** được chuyển vào bên trong Monitor (trong khối synchronized) thì điều gì sẽ xảy ra? * (1 Điểm)

01:04:16

Writer	Reader

void writeDB() {
synchronized (object) {
while (numReader > 0 ||
numWriter > 0)
object.wait();
numWriter = 1;
}
<< GHI DỮ LIỆU VÀO DB >>
(KHÔNG Ở TRONG MONITOR);
 	
synchronized (object) {
numWriter = 0;
object.notifyAll();
}
}
	void readDB() {
	synchronized (object) {
	while (numWriter > 0
	object.wait();
	numReader++;
	}
	<< ĐỌC DỮ LIỆU TỪ D
	(KHÔNG Ở TRONG MO
	
	synchronized (object) {
	numReader--;
	object.notify();
	}
	}

- [ ] Thuật toán vẫn hoạt động đúng, thoả mãn 3 yêu cầu đồng bộ của bài toán Người đọc - Người ghi
- [x] Việc này sẽ khiến cho việc sử dụng các biến chia sẻ numWriter, numReader không còn cần thiết nữa
- [x] Việc này sẽ chỉ cho phép nhiều nhất một luồng đọc được thực hiện việc đọc dữ liệu từ cơ sở dữ liệu chia sẻ tại một thời điểm bất kỳ -> không thoả mãn được yêu cầu cho phép nhiều luồng đọc cùng đọc cơ sở dữ liệu chia sẻ

13

Những phát biểu nào sau đây là đúng về khái niệm Socket? * (1 Điểm)

- [x] Socket là đối tượng được sử dụng để gửi và nhận thông điệp giữa các tiến trình trong một hệ thống phân tán
- [x] Socket cung cấp một giao diện ở mức thấp cho việc xây dựng các chương trình phân tán
- [ ] Lập trình Socket CHỈ có thể dựa trên giao thức UDP (Universal Datagram Protocol)

01:04:16

14

Thuật toán của Peterson cho bài toán loại trừ lẫn nhau trong các chương trình đồng thời có thể hoạt động đúng với bao nhiêu luồng? * (1 Điểm)

- [ ] Tối đa 3 luồng
- [ ] Tối đa 4 luồng
- [ ] Số lượng luồng bất kỳ lớn hơn 1
- [x] Chính xác 2 luồng

15

Những phát biểu nào sau đây là đúng về Thuật toán của Lamport cho bài toán truy cập tài nguyên chia sẻ trong một hệ thống phân tán? * (1 Điểm)

- [ ] Thuật toán KHÔNG thỏa mãn thuộc tính sự sống (liveness), tức là mỗi yêu cầu đi vào khu vực quan trọng cuối cùng phải được cấp quyền để đi vào khu vực quan trọng
- [x] Một tiến trình P_i nhận thấy nó có thể đi vào CS nếu: $\forall j \neq i: (q[i],i) < (q[j],j) \wedge (q[i],i) < (v[j],j)$, trong đó q[i], q[j]: dấu thời gian của yêu cầu đi vào CS của hai tiến trình P_i, P_j và v[j] là dấu thời gian của thông điệp ack từ tiến trình P_j được ghi nhận ở tiến trình P_i
- [x] Thuật toán sử dụng 3*(N-1) thông điệp cho mỗi lần truy cập khu vực quan trọng, với N là số lượng tiến trình, bao gồm: N - 1 thông điệp request, N - 1 thông điệp ack, N - 1 thông điệp release
- [x] Thuật toán đảm bảo rằng các tiến trình đi vào khu vực quan trọng theo thứ tự dấu thời gian của yêu cầu

01:04:16

Trạng thái đua tranh (race condition) giữa các luồng dẫn đến những điều nào sau đây?
* (1 Điểm)

- [x] Tính chính xác của chương trình bị phụ thuộc vào thời gian thực thi tương đối của các sự kiện
- [x] Dữ liệu chia sẻ có thể bị mất mát khi các luồng cùng thực hiện việc thay đổi dữ liệu đó
- [ ] Giá trị của các biến chia sẻ luôn luôn nhất quán và chính xác khi các luồng thực hiện việc cập nhật các biến đó

17

Những phát biểu nào sau đây là đúng khi khái quát về các hệ thống phân tán? * (1 Điểm)

- [ ] Các tiến trình trong một hệ thống phân tán luôn luôn biết được trạng thái toàn cục của hệ thống tại bất kỳ thời điểm nào
- [ ] KHÔNG tồn tại biến chia sẻ giữa các tiến trình trong một hệ thống phân tán
- [ ] Mỗi tiến trình trong hệ thống phân tán luôn CHỈ gồm có một luồng
- [x] Các tiến trình trong một hệ thống phân tán giao tiếp với nhau bằng cách gửi và nhận thông điệp qua mạng truyền thông

18

Nhược điểm chung của các thuật toán Peterson, Bakery khi sử dụng để giải quyết bài toán loại trừ lẫn nhau trong các chương trình đồng thời là gì? * (1 Điểm)

- [x] Các luồng phải liên tục kiểm tra xem điều kiện đi vào khu vực quan trọng đã được thoả mãn hay chưa, thông qua vòng lặp. Điều này dẫn đến gây lãng phí chu trình CPU
- [ ] Sử dụng các biến chia sẻ, dẫn đến có thể mất mát dữ liệu
- [ ]

01:04:16

19

Trong bài toán Người đọc và Người ghi, với n > 1 luồng đọc và m > 1 luồng ghi cùng hoạt động, chúng ta cần đảm bảo những điều kiện đồng bộ nào? * (1 Điểm)

- [x] Nhiều luồng đọc có thể đồng thời truy cập CSDL chia sẻ
- [x] Rằng buộc đọc-ghi: Một luồng đọc và một luồng ghi không được truy cập đồng thời vào CSDL chia sẻ
- [x] Rằng buộc ghi-ghi: Hai luồng ghi không được truy cập đồng thời vào CSDL chia sẻ
- [ ] Rằng buộc đọc-đọc: Hai luồng đọc không được truy cập đồng thời vào CSDL chia sẻ

20

Cho đoạn mã giả của một chương trình đồng thời với luồng *t, u* như Hình dưới đây.

Giả sử các câu lệnh được thực thi một cách nguyên tử. Sau khi hai luồng *t, u* thực thi xong các câu lệnh của mình, biến chia sẻ **counter** có thể nhận những giá trị nào sau đây? * (1 Điểm)

01:04:16

int counter = 0;

Luồng *t*	Luồng *u*

int cnt;
cnt = counter;
counter = cnt + 1;
	int cnt;
	cnt = counter;
	counter = cnt +

- [ ] 0
- [x] 2
- [ ] 3
- [x] 1

21

Những phát biểu nào sau đây là đúng về cấu trúc đồng bộ hoá Monitor? * (1 Điểm)

- [ ] Monitor KHÔNG hỗ trợ khái niệm biến điều kiện
- [x] Monitor hướng đối tượng, tức là mỗi đối tượng mặc định đi kèm với một monitor
- [x] Trong 1 thời điểm, chỉ có nhiều nhất một luồng chiếm giữ monitor

01:04:16

Giao diện **Lock** cho bài toán loại trừ lẫn nhau (mutex) trong một chương trình đồng thời gồm có những phương thức nào? * (1 Điểm)

- [x] requestCS(int pid)
- [ ] exitCS(int pid)
- [ ] enterCS(int pid)
- [x] releaseCS(int pid)

23

Trong bài toán Sản xuất và Tiêu thụ, với hai luồng **Producer** và **Consumer** cùng hoạt động, chúng ta cần đảm bảo những điều kiện đồng bộ nào? * (1 Điểm)

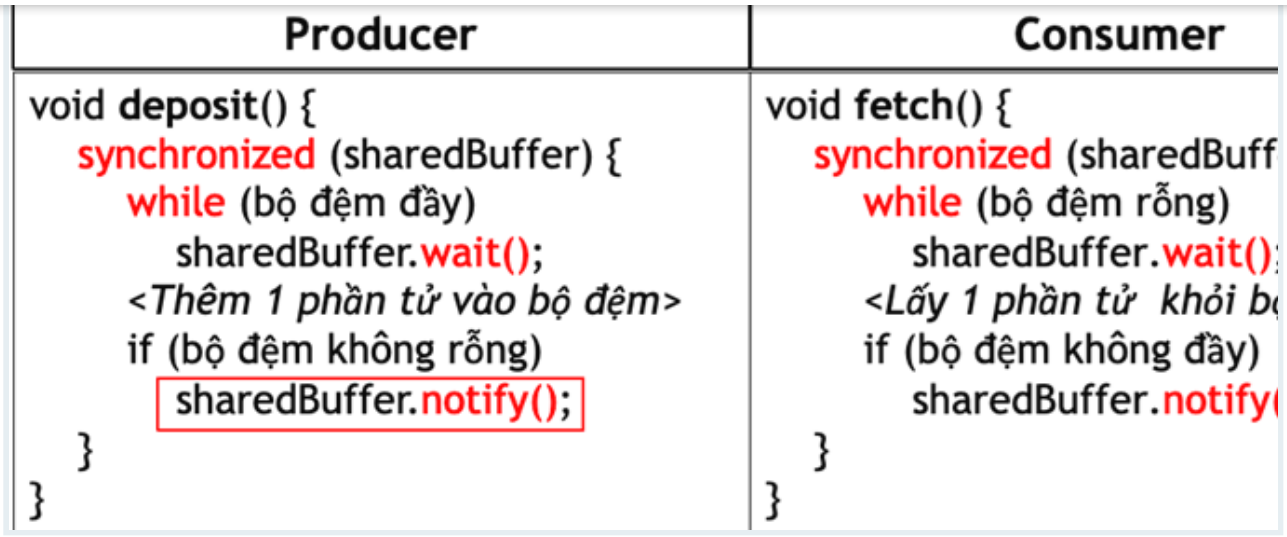
- [x] Điều kiện loại trừ lẫn nhau khi luồng Producer thực hiện việc đẩy dữ liệu vào bộ đệm và khi luồng Consumer thực hiện việc lấy dữ liệu ra khỏi bộ đệm
- [x] Điều kiện đồng bộ khi bộ đệm đầy, luồng Producer phải dừng lại
- [x] Điều kiện đồng bộ khi bộ đệm rỗng, luồng Consumer phải dừng lại
- [ ] Điều kiện đồng bộ cho phép hai luồng Producer và Consumer cùng thực hiện việc đẩy và lấy dữ liệu đồng thời

24

Cho đoạn mã giả của chương trình đa luồng để giải bài toán Sản xuất và Tiêu thụ, sử dụng monitor trong Java, như Hình dưới.

Câu lệnh gọi **notify**() của đối tượng sharedBuffer trong phương thức **deposit**(), dùng với mục đích gì? * (1 Điểm)

01:04:16



Producer	Consumer

void deposit() {
synchronized (sharedBuffer) {
while (bộ đệm đầy)
sharedBuffer.wait();
<Thêm 1 phần tử vào bộ đệm>
if (bộ đệm không rỗng)
sharedBuffer.notify();
}
}
	void fetch() {
	synchronized (sharedBuff
	while (bộ đệm rỗng)
	sharedBuffer.wait()
	<Lấy 1 phần tử khỏi b
	if (bộ đệm không đầy)
	sharedBuffer.notify
	}
	}

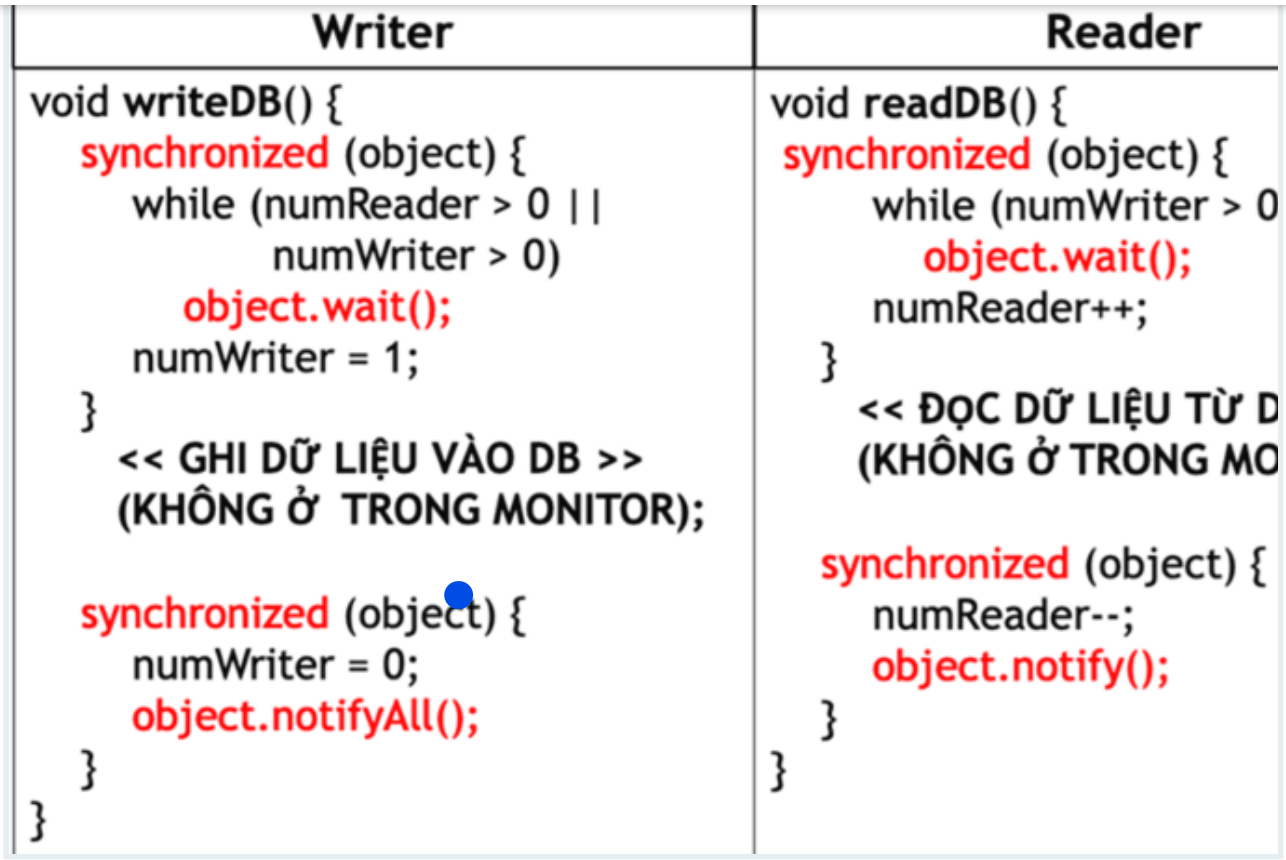
- [ ] Dùng để đánh thức cả 2 luồng Sản xuất và Tiêu thụ
- [ ] Dùng để đánh thức luồng Sản xuất (Producer) và tiếp tục công việc của nó
- [x] Dùng để đánh thức luồng Tiêu thụ (Consumer) và tiếp tục công việc của nó

25

Cho đoạn mã giả sau, cài đặt giải pháp, sử dụng cấu trúc đồng bộ hoá Monitor, cho bài toán Người đọc - Người ghi, trong đó có n > 2 luồng đọc và m > 2 luồng ghi, cùng tương tác với cơ sở dữ liệu chia sẻ.

Nếu các khối lệnh cho việc **<< GHI DỮ LIỆU VÀO DB>>** và **<< ĐỌC DỮ LIỆU TỪ DB>>** được chuyển vào bên trong Monitor (trong khối synchronized) thì điều gì sẽ xảy ra? * (1 Điểm)

01:04:16



Writer

```
void writeDB() {
   synchronized (object) {
      while (numReader > 0 ||
             numWriter > 0)
         object.wait();
      numWriter = 1;
   }
     << GHI DỮ LIỆU VÀO DB >>
     (KHÔNG Ở  TRONG MONITOR);

   synchronized (object) {
      numWriter = 0;
      object.notifyAll();
   }
}
```

Reader

```
void readDB() {
 synchronized (object) {
      while (numWriter > 0
         object.wait();
      numReader++;
   }
     << ĐỌC DỮ LIỆU TỪ D
     (KHÔNG Ở TRONG MO

   synchronized (object) {
      numReader--;
      object.notify();
   }
}
```

- [x] Việc này sẽ chỉ cho phép nhiều nhất một luồng đọc được thực hiện việc đọc dữ liệu từ cơ sở dữ liệu chia sẻ tại một thời điểm bất kỳ -> không thoả mãn được yêu cầu cho phép nhiều luồng đọc cùng đọc cơ sở dữ liệu chia sẻ
- [x] Việc này sẽ khiến cho việc sử dụng các biến chia sẻ numWriter, numReader không còn cần thiết nữa
- [ ] Thuật toán vẫn hoạt động đúng, thoả mãn 3 yêu cầu đồng bộ của bài toán Người đọc - Người ghi

26

Thuật toán Bakery của Lamport cho bài toán loại trừ lẫn nhau trong các chương trình đồng thời có thể hoạt động đúng với bao nhiêu luồng? * (1 Điểm)

- [ ] Tối đa 3 luồng
- [x] Số lượng luồng bất kỳ lớn hơn 1
- [ ] Chính xác 2 luồng

01:04:16

27

Những phát biểu nào sau đây là đúng về RMI (Remote Method Invocations)? * (1 Điểm)

- [x] Trong RMI, tiến trình gửi yêu cầu không cần biết đến vị trí thực sự của đối tượng từ xa
- [ ] Hai đối tượng stub và skeleton KHÔNG bắt buộc phải được tạo ra khi thực hiện một lời gọi từ xa
- [x] Trong RMI, chúng ta có thêm một thành phần trung gian RMIRegistry, giúp đăng ký và lấy về tham chiếu các đối tượng từ xa
- [x] Mỗi lời gọi từ xa luôn được thực hiện thông qua hai đối tượng đại diện: stub ở phía client và skeleton ở phía server

28

Những phát biểu nào sau đây là đúng về MOM (Message-Oriented Middleware)? * (1 Điểm)

- [x] Trong MOM, luôn có một thành phần trung gian, được gọi là Messaging Server, để điều phối quá trình gửi và nhận các thông điệp
- [ ] Để thực hiện được quá trình truyền thông điệp, hai tiến trình gửi và nhận bắt buộc phải thực thi tại cùng một thời điểm
- [x] Các hệ thống dựa trên MOM cho phép việc truyền thông diễn ra thông qua trao đổi bất đồng bộ các thông điệp

29

Trong các chương trình đồng thời, hay chương trình đa luồng, các luồng giao tiếp với nhau, chủ yếu, bằng cách nào? * (1 Điểm)

01:04:16

- [x] Thông qua các biến chia sẻ

30

Các tiến trình (Process) trong một hệ thống phân tán gồm nhiều máy tính đặt tại nhiều địa điểm khác nhau giao tiếp với nhau bằng cách nào ? * (1 Điểm)

- [ ] Sử dụng mô hình bộ nhớ chia sẻ
- [ ] Không sử dụng mô hình bộ nhớ chia sẻ, cũng như mô hình truyền thông điệp qua mạng truyền thông
- [x] Sử dụng mô hình truyền thông điệp qua mạng truyền thông

31

Các luồng (Thread) trong một hệ thống đồng thời dựa trên cơ chế khoá, chạy trên một máy tính gồm nhiều bộ vi xử lý, giao tiếp với nhau bằng cách nào ? * (1 Điểm)

- [x] Sử dụng mô hình bộ nhớ chia sẻ
- [ ] Không sử dụng mô hình bộ nhớ chia sẻ, cũng như mô hình truyền thông điệp qua mạng truyền thông
- [ ] Sử dụng mô hình truyền thông điệp qua mạng truyền thông

32

Xét một giải pháp cho bài toán Sản xuất và Tiêu thụ, sử dụng cấu trúc đồng bộ hoá Semaphore, như Hình dưới.

Dòng lệnh **isEmpty.V();** trong phương thức **deposit()** được dùng với mục đích gì? * (1 Điểm)

01:04:16

CountingSemaphore isFull(size), isEmpty(0)

Producer	Consumer

void deposit() {
isFull.P();
mutex.P();
<Ghi dữ liệu vào bộ đệm>
mutex.V();
isEmpty.V();
}
	void fetch() {
	isEmpty.P();
	mutex.P();
	<Lấy dữ liệu ra khỏi bộ
	mutex.V();
	isFull.V();
	}

- [ ] Đánh thức luồng Producer dậy để thực thi tiếp công việc của nó
- [x] Đánh thức luồng Consumer dậy để thực thi tiếp công việc của nó
- [ ] Khoá luồng Consumer
- [ ] Khoá luồng Producer

33

Những phát biểu nào sau đây là đúng về hai giao thức truyền thông điệp: UDP và TCP? * (1 Điểm)

- [ ] TCP KHÔNG đảm bảo thứ tự nhận được của các gói tin giống như thứ tự đã gửi
- [x] TCP là một giao thức kết nối đáng tin cậy, tức là không bị mất mát gói tin trên đường truyền
- [x] Các gói tin được gửi theo giao thức UDP có thể bị mất trên đường truyền
- [x] Các gói tin được gửi theo giao thức UDP KHÔNG được bảo đảm nhận được theo thứ tự đã gửi

01:04:16

Số lượng khu vực quan trọng (CS) của một luồng, trong các chương trình đồng thời, có thể là bao nhiêu? * (1 Điểm)

- [x] Không giới hạn
- [ ] 3
- [ ] 1
- [ ] 2
- [ ] 0

35

Tổng số luồng được tạo ra trong chương trình đồng thời sau là bao nhiêu, **nếu không tính luồng main**? * (1 Điểm)

01:04:16

```
    int result;

    public Fibonacci(int n) {
        this.n = n;
    }

    public void run() {
        if ((n == 0)||(n == 1 )) result = 1;
        else {
            Fibonacci f1 = new Fibonacci(n-1);
            Fibonacci f2 = new Fibonacci(n-2);
            f1.start();
            f2.start();
            try {
                f1.join();
                f2.join();
            } catch (InterruptedException e){
            };
            result = f1.getResult() + f2.getResult();
        }
    }

    public int getResult(){
        return result;
    }

    public static void main(String[] args) {
        Fibonacci f1 = new Fibonacci(3);
        f1.start();
        try {
            f1.join();
        } catch (InterruptedException e){};
        System.out.println("So Fibonacci tuong ung la: " + f1.ge
    }
}
```

- [ ] 5
- [x] 6
- [ ] 4
- [ ] 9
- [ ] 3
- [ ] 7

01:04:16

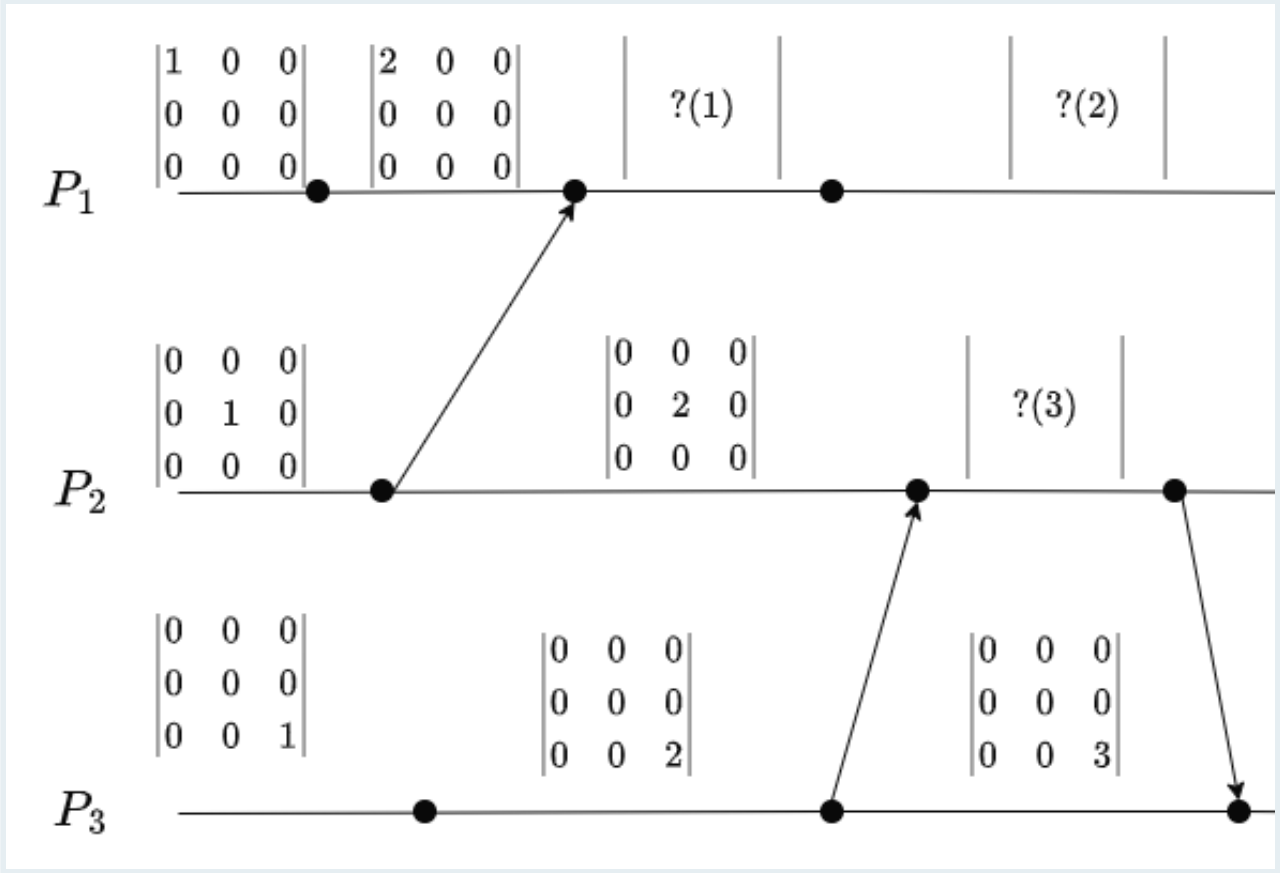
36

Những phát biểu nào sau đây là đúng về bài toán truy cập tài nguyên chia sẻ, hay bài toán loại trừ lẫn nhau, trong hệ thống phân tán? * (1 Điểm)

- [ ] Do không có bộ nhớ chia sẻ giữa các tiến trình trong hệ thống phân tán, nên không thể sử dụng các cấu trúc đồng bộ hoá như Semaphore hoặc Monitor cho bài toán này
- [x] Có thể sử dụng Semaphore cho bài toán truy cập tài nguyên chia sẻ trong một hệ thống phân tán
- [x] Có thể sử dụng Monitor cho bài toán truy cập tài nguyên chia sẻ trong một hệ thống phân tán

37

Hãy xác định ma trận tương ứng (1), (2), (3), (4) trong đồng hồ ma trận bên dưới * (1 Điểm)



01:04:16

Error]

- [ ] *[Math Processing Error]*

- [ ] *[Math Processing Error]*

38

Khi chạy chương trình sau có thể sinh ra những kết quả nào sau đây? * (1 Điểm)

```
public class Lab1 extends Thread{
    private int id;
    public Lab1(int _id) {
        id = _id;
    }
    public void run() {
            System.out.print("T-" + id + " ");
    }

    public static void main(String[] args) throws InterruptedE
        Lab1 t1 = new Lab1(1);
        Lab1 t2 = new Lab1(2);
        t1.start();
        t2.start();

        t1.join();

        System.out.print("T-m ");


    }
}
```

- [ ] T-2 T-m T-1

- [ ] T-m T-1 T-2

- [x] T-1 T-2 T-m

01:04:16

- [x] T-2 T-1 T-m
- [x] T-1 T-m T-2

39

Ưu điểm của việc sử dụng Semaphore cho bài toán loại trừ lẫn nhau trong các chương trình đồng thời, so với các thuật toán Peterson, Dekker, Bakery là gì? * (1 Điểm)

- [x] Thời gian chạy chương trình nhanh hơn
- [x] Giải quyết được vấn đề bận chờ (busy-waiting), không gây lãng phí chu trình CPU
- [ ] Không có ưu điểm gì hơn so với các thuật toán đó

Gửi