Quizizz

NAME : _____

CLASS : _____

Chương 6 - 7
30 Questions

DATE : _____

1. Đoạn mã nào trong các tiến trình có thể gây ra lỗi khi được thực thi đồng thời?

A Entry Section.

B Critical Section.

C Exit Section.

D Remainer Section.

2. Đồng bộ hoá (Process Synchronization) là công việc cần phải áp dụng cho loại tiến trình nào?

A Tiến trình người dùng (User process).

B Tiến trình cộng tác (Cooperating process).

C Tiến trình hệ thống (System process).

D Tiến trình độc lập (Independent process).

3. Đoạn mã nào được sử dụng để kiểm soát quá trình đồng bộ?

A Critical section.

B Program code.

C Entry section.

D Remainder section.

4. Đoạn mã nào có thể chạy cùng lúc mà không gây ra sai sót dữ liệu?

A Remainder section.

B Program code.

C Critical section.

D Entry section.

5. Biến số đơn nguyên (atomic varible) là gì?

A Biến số chỉ có ý nghĩa địa phương, sử dụng nội bộ trong tiểu trình.

B Các thao tác lên biến số này tuần tự được thực thi trong CPU.

C Các thao tác lên biến số này được song song thực hiện trong CPU.

D Biến số chỉ chứa duy nhất một kiểu dữ liệu được định nghĩa trước.

6. Giải thuật Peterson sử dụng các biến số điều khiển nào để giải quyết bài toán đồng bộ giữa hai tiến trình?

A int sync = 2;

B boolean flag[2];

C choosing[i] = true và int number[i];

D boolean flag[2] và int turn;

7. Một tiến trình Px thực hiện thao tác signal() trên một biến số Semaphore n thì có tác dụng gì?

A n++ và sau đó nếu n > 0 thì wake_up() một tiến trình đang bị blocked.

B n++ và sau đó nếu n <= 0 thì wake_up() tiến trình Px.

C n++ và sau đó nếu n > 0 thì wake_up() tiến trình Px.

D n++ và sau đó nếu n <= 0 thì wake_up() tiến trình đang bị blocked.

8. Một tiến trình Px thực hiện thao tác wait() trên một biến số Semaphore n thì có tác dụng gì?

A n-- và sau đó nếu n < 0 thì block() tiến trình Px.

B n-- và sau đó nếu n <= 0 thì block() các tiến trình khác Px.

C n++ và sau đó nếu n <= 0 thì block() tiến trình Px.

D n-- và sau đó nếu n >= 0 thì block() tiến trình Px.

9. Giải thuật / Phương pháp nào sau đây chỉ có thể giải quyết đồng bộ không nhiều hơn 2 tiến trình?

A Phương pháp Hàng rào bộ nhớ.

B Giải thuật Peterson.

C Giải thuật Banker.

D Phương pháp Semaphore.

10. Tình trạng cạnh tranh (Race condition) là gì?

A Tiến trình không cho phép các tiến trình khác tác động lên biến số của nó, và dẫn đến việc đồng bộ thất bại.

B Các lệnh cấp thấp (là mã máy) được thực thi đồng thời trong một chu kỳ lệnh của CPU làm sai sót dữ liệu.

C Người sử dụng yêu cầu chạy 02 tiến trình có tranh chấp dữ liệu, gây nên hiện tượng tắc nghẽn cho hệ thống.

D Khi nhiều hơn một tiến trình thao tác lên dữ liệu chia sẻ, kết quả cuối cùng phụ thuộc vào thứ tự thực thi của các thao tác đó.

11. Kỹ thuật đồng bộ sử dụng Semaphore giải quyết được vấn đề gì mà giải thuật Peterson chưa làm được?

A Mutual Exclusion (Loại trừ tương hỗ).

B Progress (Tính tiến triển).

C Bounded-Waiting (Chờ vô hạn định).

D Busy-waiting (Chờ đợi bận rộn).

12. Yêu cầu về tính sống còn (liveness) của các giải pháp đồng bộ đảm bảo điều gì cho hệ thống?

A Dữ liệu luôn được đồng bộ và không có sai sót khi cập nhật.

B Hệ thống đang xử lý các tiến trình có hiệu năng khai thác cao.

C Các tiến trình luôn tiến triển, tài nguyên không cạn kiệt.

D Sự chờ đợi bận rộn (Busy waiting) không xuất hiện với mọi tiến trình.

13. Mục đích của việc sử dụng Semaphore là gì?

A Trị số của Semaphore cho biết process nào đang được thực thi.

B Semaphore là tín hiệu ngắt gửi cho hệ điều hành khi cần đồng bộ tiến trình.

C Thông tin của Semaphore phục vụ cho bài toán đồng bộ tiến trình.

D Trị số của Semaphore cho biết số tiến trình tối đa được vào hệ thống.

14. Phương pháp Hàng rào bộ nhớ (Memory Barrier) được hiện thực ra sao?

A Việc cập nhật vùng nhớ chia sẻ được quyết định bởi tiến trình cấp phát hàng rào.

B Các câu lệnh thay đổi biến số chia sẻ cần được nhìn thấy bởi mọi tiến trình khác.

C Các vùng nhớ chia sẻ cần được nhìn thấy bởi tất cả tiến trình đang đồng bộ.

D Các tiến trình được cấp các bản sao vùng nhớ chia sẻ để thao tác cập nhật.

15. “Critical Section” mô tả đoạn mã như thế nào trong một tiến trình?

A Đoạn mã có yêu cầu nhập xuất dữ liệu từ thiết bị ngoại vi.

B Đoạn mã có yêu cầu tính toán và sử dụng toàn bộ CPU.

C Đoạn mã có chứa những thao tác lên biến dùng chung.

D Đoạn mã hệ điều hành tự thêm vào trong tiến trình.

16. "Entry / Exit Section" là đoạn mã gì?

A. Đoạn mã hệ điều hành thêm vào trước và sau đoạn mã nguy cơ (Critical section).

B. Đoạn mã có chứa lệnh can thiệp vào hoạt động của hệ điều hành.

C. Đoạn mã có chứa những thao tác lên biến dùng chung.

D. Đoạn mã có yêu cầu tính toán và sử dụng toàn bộ CPU.

17.

Time	P1	P2
t = 1	wait(S)	
t = 2		wait(Q)
t = 3	wait(Q)	
t = 4		wait(S)
t = 5	signal(S)	
t = 6		signal(Q)
t = 7	signal(Q)	
t = 8		signal(S)

Cho hai tiến trình P1 và P2 quyền tác động lên biến semaphore chia sẻ S và Q (đều có khởi tạo = 1). Các lệnh sau đây lần lượt được thực thi, hệ thống sẽ diễn tiến như thế nào?

A. Hệ thống sẽ rơi vào trạng thái Deadlock.

B. Hệ thống sẽ đảm bảo P2 hoàn tất trước P1.

C. Hệ thống sẽ chạy hết tất cả lệnh đã nêu.

D. Hệ thống sẽ đảm bảo P1 hoàn tất trước P2.

18. Cho đoạn mã của 2 tiến trình như sau:

```
P1:
(các lệnh khác)
wait(mutex);
   critical section
signal(mutex);
(các lệnh khác)
```

```
P2:
(các lệnh khác)
wait(mutex);
   critical section
signal(mutex);
(các lệnh khác)
```

Trong đó biến mutex là biến toàn cục dùng chung (shared variable). Phát biểu nào sau đây là đúng với hệ thống nêu trên?

A. Với khởi tạo mutex = 1; chỉ có 1 tiến trình được vào critical section.

B. Với khởi tạo mutex = 1; P2 phải gửi tín hiệu đến P1 để xin vào critical section.

C. Với khởi tạo mutex = 0; chỉ có 1 tiến trình được vào critical section.

D. Với khởi tạo mutex = 2; P2 chắc chắn sẽ vào critical section trước.

19. Cho đoạn mã của 2 tiến trình P1 và P2 như sau:

```
P1:
(các lệnh khác)
signal(mutex);
func_1();
(các lệnh khác)
```

```
P2:
(các lệnh khác)
wait(mutex);
func_2();
(các lệnh khác)
```

Trong đó biến mutex là biến toàn cục dùng chung (Shared variable). Chọn phát biểu đúng:

A. Để đảm bảo hàm func_1() chạy trước func_2(), khởi tạo mutex = 0.

B. Để đảm bảo hàm func_2() chạy trước func_1(), khởi tạo mutex = 2.

C. Để đảm bảo hàm func_2() chạy trước func_1(), khởi tạo mutex = 0.

D. Để đảm bảo hàm func_1() chạy trước func_2(), khởi tạo mutex = 1.

20. Semaphore được hiện thực như thế nào?

A Mảng các số nguyên hoặc nhị phân, kèm theo 2 thao tác block() và wake_up().

B Mảng các số nguyên hoặc nhị phân, kèm theo 2 thao tác wait() và signal().

C Biến số nguyên hoặc nhị phân, kèm theo 2 thao tác block() và wake_up().

D Biến số nguyên hoặc nhị phân, kèm theo 2 thao tác wait() và signal().

21. Bài toán "Bộ đếm giới hạn" (Bounded Buffer) đề cập vấn đề chính yếu gì?

A Bảo mật thông tin khi gửi và nhận thông điệp giữa các tiến trình

B Khóa chặn truy cập chỉ của một vài tiến trình đang được thực thi.

C Gửa và nhận gói tin qua bộ nhớ chia sẻ có kích thước nhất định.

D Tranh chấp tài nguyên giữa nhiều tiến trình trong lúc thực thi

22. Bài toán "Bộ ghi - Bộ đọc" (Writers and Readers) đề cập đến vấn đề chính yếu gì?

A Bảo mật thông tin khi chia sẻ thông tin giữa các tiến trình đang thực thi.

B Gửi và nhận gói tin qua bộ nhớ chia sẻ có kích thước nhất định.

C Phân phối dữ liệu từ nhiều tiến trình nguồn đến nhiều tiến trình đích.

D Dữ liệu chia sẻ mà chỉ có một vài tiến trình mới có nhu cầu cập nhật dữ liệu.

23. Bài toán "Triết gia ăn tối" (Dining Philosophers) đề cập đến vấn đề chính yếu gì?

A Hiệu suất sử dụng tài nguyên trong hệ thống chạy song song nhiều tiến trình

B Tranh chấp các tài nguyên chia sẻ riêng biệt giữa từng cặp tiến trình

C Bảo mật thông tin chia sẻ thông tin giữa nhiều tiến trình với nhau

D Chia sẻ tài nguyên thành nhiều thực thể để đáp ứng cho nhiều tiến trình.

24. Bài toán "Bộ đếm giới hạn" (Bounded Buffer) có thể giải quyết bằng bao nhiêu biến số Semaphore?

A 3 biến: mutex, full và empty.

B 1 mảng sem[5].

C Chỉ cần 2 biến: full và empty

D Duy nhất một biến số n

25. Bài toán "Bộ ghi - Bộ đọc" (Writers and Readers) có đặc trưng gì?

A. Hệ thống chỉ có một bộ ghi và rất nhiều bộ đọc

B. Hệ thống chỉ có một bộ đọc và rất nhiều bộ ghi

C. Các bộ đọc mới có thể cập nhật dữ liệu chia sẻ

D. Tất cả bộ đọc và bộ ghi cần xếp hàng để thực thi

26. Bài toán "Triết gia ăn tối" (Dining Philosophers) nếu sử dụng semaphore thì chúng được khởi tạo như thế nào?

A. semaphore chopstick[5], tất cả phần từ gán bằng 1.

B. semaphore chopstick, khởi tạo giá trị 5.

C. semaphore chopstick[5], tất cả phần tử gán bằng 1.

D. semaphore chopstick[5], các phần tử gán lần lượt từ 1 đến 5.

27. API POSIX cung cấp nhiều công cụ đồng bộ, nhưng không bao gồm công cụ nào sau đây?

A. Biến số điều kiện(condition variable).

B. Biến số semaphore.

C. Khóa mutex clock.

D. Dispatcher objects.

28. Bài toán "Bộ ghi - Bộ đọc" (Writers and Readers) các biến số được khởi tạo như thế nào?

A. semaphore rw_mutex = 1, mutex = 1; int read_count = 2;

B. semaphore rw_mutex = 1, mutex = 1; int read_count = 0;

C. semaphore rw_mutex = 0, mutex = 1; int read_count = 0;

D. semaphore rw_mutex = 1, mutex = 2; int read_count = 0;

29. Bài toán "Triết gia ăn tối" (Dining Philosophers) có thể giải quyết bằng phương pháp nào để tránh bị tắc nghẽn(deadlock)?

A. Các biến số semphore với các lệnh wait() và signal().

B. Các khóa mutex_lock áp dụng cho từng vùng tranh chấp.

C. Bỏ quan sát(Monitor) với các lệnh test().

D. Giải thuật Peterson với các vòng lặp kiểm tra while().

30. Bài toán "Bộ ghi - Bộ đọc" (Writers and Readers) có biến thể thứ 2, nó khác gì với biến thể đầu tiên?

A Các bộ đọc có thể thực thi song song mà không sai sót dữ liệu.

B Nếu một bộ ghi mới đến, nó sẽ được thực thi sớm nhất có thể.

C Nếu một bộ đọc mới đến, nó sẽ được thực thi sớm nhất có thể.

D Số lượng bộ đọc và bộ ghị bị giới hạn để tránh cạn kiệt tài nguyên.

Answer Key

1. b	2. b	3. c	4. a
5. b	6. d	7. d	8. a
9. b	10. d	11. d	12. c
13. c	14. b	15. c	16. a
17. a	18. a	19. a	20. d
21. c	22. d	23. b	24. a
25. c	26. c	27. d	28. b
29. c	30. b		